```matlab
classdef FUNMAP_Input_exported < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        Friction________________________________Label_9   matlab.ui.control.Label
        Panel_32                        matlab.ui.container.Panel
        Button_41                       matlab.ui.control.Button
        Mpif90EditField                 matlab.ui.control.EditField
        Mpif90EditFieldLabel            matlab.ui.control.Label
        DoublePrecisionCheckBox         matlab.ui.control.CheckBox
        IntelCompilerCheckBox           matlab.ui.control.CheckBox
        ParallelModeCheckBox            matlab.ui.control.CheckBox
        MakefileEditField               matlab.ui.control.EditField
        MakefileEditFieldLabel          matlab.ui.control.Label
        Button_42                       matlab.ui.control.Button
        StartsimulationButton           matlab.ui.control.Button
        CreateMakefileButton            matlab.ui.control.Button
        TabGroup                        matlab.ui.container.TabGroup
        InputTab                        matlab.ui.container.Tab
        Friction________________________________Label_10   matlab.ui.control.Label
        Panel_33                        matlab.ui.container.Panel
        SteadyTimeEditField             matlab.ui.control.NumericEditField
        SteadyTimeEditFieldLabel        matlab.ui.control.Label
        TimeintervalsecEditField        matlab.ui.control.NumericEditField
        TimeintervalsecEditFieldLabel   matlab.ui.control.Label
        UseSmagorinskyCheckBox          matlab.ui.control.CheckBox
        GenerateinputtxtButton          matlab.ui.control.Button
        Friction________________________________Label_7   matlab.ui.control.Label
        Panel_30                        matlab.ui.container.Panel
        C2EditField                     matlab.ui.control.NumericEditField
        C2EditFieldLabel                matlab.ui.control.Label
        C1EditFieldLabel                matlab.ui.control.Label
        C1EditField                     matlab.ui.control.NumericEditField
        ButtonGroup_23                  matlab.ui.container.ButtonGroup
        ViscositybreakingButton         matlab.ui.control.RadioButton
        ShockwavecapturingButton        matlab.ui.control.RadioButton
        DimensionLabel                  matlab.ui.control.Label
        DepthLabel                      matlab.ui.control.Label
        Friction________________________________Label_6   matlab.ui.control.Label
        Panel_29                        matlab.ui.container.Panel
        SavefilestoEditField            matlab.ui.control.EditField
        SavefilestoLabel                matlab.ui.control.Label
        Button_14                       matlab.ui.control.Button
        Friction________________________________Label_5   matlab.ui.control.Label
        Panel_28                        matlab.ui.container.Panel
        SpacingEditField                matlab.ui.control.NumericEditField
        SpacingEditFieldLabel           matlab.ui.control.Label
        BoundarylimitsButton            matlab.ui.control.Button
        WestEditField_2                 matlab.ui.control.NumericEditField
        SouthEditField_2                matlab.ui.control.NumericEditField
        EastEditField_2                 matlab.ui.control.NumericEditField
        NorthEditField_2                matlab.ui.control.NumericEditField
        ImportgaugelistCheckBox         matlab.ui.control.CheckBox
        LoadFileEditField               matlab.ui.control.EditField
        Button_7                        matlab.ui.control.Button
        CreatenestedgridboundaryCheckBox  matlab.ui.control.CheckBox
        BottomFrictionLabel             matlab.ui.control.Label
        Panel_26                        matlab.ui.container.Panel
        ConstantFrictionValue           matlab.ui.control.NumericEditField
        ValueEditFieldLabel             matlab.ui.control.Label
        Button_10                       matlab.ui.control.Button
        FrictionFileEditField           matlab.ui.control.EditField
        FrictionFileEditFieldLabel      matlab.ui.control.Label
        ButtonGroup_8                   matlab.ui.container.ButtonGroup
        NonconstantButton               matlab.ui.control.RadioButton
        ConstantCoefficientButton       matlab.ui.control.RadioButton
        PhysicsLabel                    matlab.ui.control.Label
        ObstaclesLabel                  matlab.ui.control.Label
        Panel_24                        matlab.ui.container.Panel
        AddobstaclesCheckBox            matlab.ui.control.CheckBox
        ObstacleFilename                matlab.ui.control.EditField
        ObstacleLoadFile                matlab.ui.control.Button
        SpongeBoundaryLabel             matlab.ui.control.Label
        Panel_23                        matlab.ui.container.Panel
        MaximumCspEditField             matlab.ui.control.NumericEditField
        MaximumCspEditFieldLabel        matlab.ui.control.Label
        MaximumCdEditField              matlab.ui.control.NumericEditField
        MaximumCdEditFieldLabel         matlab.ui.control.Label
        DecayTypeDropDown               matlab.ui.control.DropDown
        DecayTypeDropDownLabel          matlab.ui.control.Label
        SouthLabel                      matlab.ui.control.Label
        WestLabel                       matlab.ui.control.Label
        EastLabel                       matlab.ui.control.Label
        NorthLabel                      matlab.ui.control.Label
        WestEditField                   matlab.ui.control.NumericEditField
        SouthEditField                  matlab.ui.control.NumericEditField
        EastEditField                   matlab.ui.control.NumericEditField
        NorthEditField                  matlab.ui.control.NumericEditField
        DiffusionCheckBox               matlab.ui.control.CheckBox
        FrictionCheckBox                matlab.ui.control.CheckBox
        DirectCheckBox                  matlab.ui.control.CheckBox
        LayerwidthLabel                 matlab.ui.control.Label
        GridLabel                       matlab.ui.control.Label
        Panel_22                        matlab.ui.container.Panel
        GridSizeLabel                   matlab.ui.control.Label
        SouthwestcornerLabel            matlab.ui.control.Label
        ButtonGroup_4                   matlab.ui.container.ButtonGroup
        Button_2                        matlab.ui.control.RadioButton
        CartesianButton                 matlab.ui.control.RadioButton
        SphericalButton                 matlab.ui.control.RadioButton
        LatitudeEditField               matlab.ui.control.NumericEditField
        LatitudeEditFieldLabel          matlab.ui.control.Label
        LongitudeEditField              matlab.ui.control.NumericEditField
        LongitudeEditFieldLabel         matlab.ui.control.Label
        YEditField                      matlab.ui.control.NumericEditField
        YgridsizeLabel                  matlab.ui.control.Label
        XEditField                      matlab.ui.control.NumericEditField
        XEditFieldLabel                 matlab.ui.control.Label
        TimeLabel_2                     matlab.ui.control.Label
        Panel 21                        matlab.ui.container.Panel
```

```
    TotalEditField                  matlab.ui.control.NumericEditField
    TotalEditFieldLabel             matlab.ui.control.Label
    SimulationIntervalEditField     matlab.ui.control.NumericEditField
    SimulationIntervalEditFieldLabel  matlab.ui.control.Label
    GaugeIntervalEditField          matlab.ui.control.NumericEditField
    GaugeIntervalEditFieldLabel     matlab.ui.control.Label
    StartEditField                  matlab.ui.control.NumericEditField
    ScreenLoadingIntervalLabel      matlab.ui.control.Label
    PrintLabel                      matlab.ui.control.Label
    Panel_20                        matlab.ui.container.Panel
    ResultsfolderEditField          matlab.ui.control.EditField
    ResultsfolderEditFieldLabel     matlab.ui.control.Label
    Panel_19                        matlab.ui.container.Panel
    XEditField_2                    matlab.ui.control.NumericEditField
    XEditField_2Label               matlab.ui.control.Label
    YEditField_2                    matlab.ui.control.NumericEditField
    YEditField_2Label               matlab.ui.control.Label
    Panel_18                        matlab.ui.container.Panel
    AngleEditField                  matlab.ui.control.NumericEditField
    AngleEditFieldLabel             matlab.ui.control.Label
    StartatXEditField               matlab.ui.control.NumericEditField
    StartatXEditFieldLabel          matlab.ui.control.Label
    DepthmEditFieldLabel            matlab.ui.control.Label
    DepthmEditField                 matlab.ui.control.NumericEditField
    InvertvaluesCheckBox            matlab.ui.control.CheckBox
    WaterlevelmEditFieldLabel       matlab.ui.control.Label
    WaterlevelmEditField            matlab.ui.control.NumericEditField
    Button_8                        matlab.ui.control.Button
    DepthFileEditField              matlab.ui.control.EditField
    ButtonGroup_7                   matlab.ui.container.ButtonGroup
    Button4                         matlab.ui.control.RadioButton
    SlopeButton                     matlab.ui.control.RadioButton
    FlatButton                      matlab.ui.control.RadioButton
    ImportDataButton                matlab.ui.control.RadioButton
    Friction________________________________Label_2  matlab.ui.control.↙
Label
    ParallelInfoLabel               matlab.ui.control.Label
    Panel_17                        matlab.ui.container.Panel
    Panel_5                         matlab.ui.container.Panel
    ProcessorY                      matlab.ui.control.NumericEditField
    YLabel                          matlab.ui.control.Label
    ProcessorX                      matlab.ui.control.NumericEditField
    ProcessornumbersXLabel          matlab.ui.control.Label
    Panel_25                        matlab.ui.container.Panel
    ButtonGroup_3                   matlab.ui.container.ButtonGroup
    FroudeCapEditField              matlab.ui.control.NumericEditField
    FroudeCapEditFieldLabel         matlab.ui.control.Label
    NonlinearShallowWaterEquationButton  matlab.ui.control.RadioButton
    Button                          matlab.ui.control.RadioButton
    FullyNonlinearBoussinesqEquationButton  matlab.ui.control.RadioButton
    LinearShallowWaterEquationButton  matlab.ui.control.RadioButton
    Panel_27                        matlab.ui.container.Panel
    BottomFrictionEditField         matlab.ui.control.NumericEditField
    BottomFrictionEditFieldLabel    matlab.ui.control.Label
    WetDrySchemeEditField           matlab.ui.control.NumericEditField
    WetDrySchemeEditFieldLabel      matlab.ui.control.Label
    InitialConditionTab             matlab.ui.container.Tab
    SedimentTransportCheckBox       matlab.ui.control.CheckBox
```

```
        GenerateinputtxtButton_4        matlab.ui.control.Button
        ButtonGroup_18                  matlab.ui.container.ButtonGroup
        VesselButton                    matlab.ui.control.RadioButton
        MeteotsunamiButton              matlab.ui.control.RadioButton
        Button_31                       matlab.ui.control.RadioButton
        WavemakerButton                 matlab.ui.control.RadioButton
        ImportUVZButton                 matlab.ui.control.RadioButton
        SelectInitialConditonLabel      matlab.ui.control.Label
        TabGroup2                       matlab.ui.container.TabGroup
        Tab_5                           matlab.ui.container.Tab
        Tab                             matlab.ui.container.Tab
        Button_33                       matlab.ui.control.Button
        VVelocityEditField              matlab.ui.control.EditField
        VVelocityEditFieldLabel         matlab.ui.control.Label
        UVelocityEditField              matlab.ui.control.EditField
        UVelocityEditFieldLabel         matlab.ui.control.Label
        SurfaceHeightEditField          matlab.ui.control.EditField
        SurfaceHeightEditFieldLabel     matlab.ui.control.Label
        Button_34                       matlab.ui.control.Button
        Button_32                       matlab.ui.control.Button
        ImportUVZfilesLabel             matlab.ui.control.Label
        Label_9                         matlab.ui.control.Label
        Tab_2                           matlab.ui.container.Tab
        WaveHeightmEditField            matlab.ui.control.NumericEditField
        WaveHeightmEditFieldLabel       matlab.ui.control.Label
        MaximumEditField                matlab.ui.control.NumericEditField
        MaximumEditFieldLabel           matlab.ui.control.Label
        MinimumEditField                matlab.ui.control.NumericEditField
        MinimumEditFieldLabel           matlab.ui.control.Label
        PeakEditField                   matlab.ui.control.NumericEditField
        FrequencyPeakLabel              matlab.ui.control.Label
        FrequencyLabel                  matlab.ui.control.Label
        WaveCompCountEditField          matlab.ui.control.NumericEditField
        WavecomponentcountEditFieldLabel  matlab.ui.control.Label
        WavecomponentfileEditField      matlab.ui.control.EditField
        WavecomponentfileEditFieldLabel  matlab.ui.control.Label
        ThetadegreesEditField           matlab.ui.control.NumericEditField
        ThetadegreesEditFieldLabel      matlab.ui.control.Label
        PeriodsecEditField              matlab.ui.control.NumericEditField
        PeriodsecEditFieldLabel         matlab.ui.control.Label
        WidthDeltaEditField             matlab.ui.control.NumericEditField
        WidthDeltaEditFieldLabel        matlab.ui.control.Label
        TimerampsecEditField            matlab.ui.control.NumericEditField
        TimerampsecEditFieldLabel       matlab.ui.control.Label
        WidthmEditField                 matlab.ui.control.NumericEditField
        WidthmEditFieldLabel            matlab.ui.control.Label
        YcoordinatemEditField           matlab.ui.control.NumericEditField
        YcoordinatemEditFieldLabel      matlab.ui.control.Label
        XcoordinatemEditField           matlab.ui.control.NumericEditField
        XcoordinatemEditFieldLabel      matlab.ui.control.Label
        LagtimesecEditField             matlab.ui.control.NumericEditField
        LagtimesecEditFieldLabel        matlab.ui.control.Label
        WaterdepthmEditField            matlab.ui.control.NumericEditField
        WaterdepthmEditFieldLabel       matlab.ui.control.Label
        AmplitudemEditField             matlab.ui.control.NumericEditField
        AmplitudemEditFieldLabel        matlab.ui.control.Label
        WavemakertypeDropDown           matlab.ui.control.DropDown
        WavemakertypeDropDownLabel      matlab.ui.control.Label
        Button_20                       matlab.ui.control.Button
        WavemakerParametersLabel_2      matlab.ui.control.Label
        Label_10                        matlab.ui.control.Label
        Tab_4                           matlab.ui.container.Tab
        WindLabel                       matlab.ui.control.Label
        PressureLabel                   matlab.ui.control.Label
        Button_43                       matlab.ui.control.Button
        PressurefileEditField           matlab.ui.control.EditField
        PressurefileEditFieldLabel      matlab.ui.control.Label
        StormfileEditFieldLabel         matlab.ui.control.Label
        StormfileEditField              matlab.ui.control.EditField
        WindwaveinteractionSwitch       matlab.ui.control.Switch
        WindwaveinteractionSwitchLabel  matlab.ui.control.Label
        HollandstormmodelSwitch         matlab.ui.control.Switch
        HollandstormmodelSwitchLabel    matlab.ui.control.Label
        ConstantwindfileEditField       matlab.ui.control.EditField
        ConstantwindfileEditFieldLabel  matlab.ui.control.Label
        ConstantwindfieldSwitch         matlab.ui.control.Switch
        ConstantwindfieldSwitchLabel    matlab.ui.control.Label
        CrestRatioEditField             matlab.ui.control.NumericEditField
        CrestRatioEditFieldLabel        matlab.ui.control.Label
        WindstresscoefficientEditFieldLabel  matlab.ui.control.Label
        WindstresscoefficientEditField  matlab.ui.control.NumericEditField
        AirpressureSwitch               matlab.ui.control.Switch
        AirpressureSwitchLabel          matlab.ui.control.Label
        WindforceSwitch                 matlab.ui.control.Switch
        WindforceSwitchLabel            matlab.ui.control.Label
        Button_30                       matlab.ui.control.Button
        Button_29                       matlab.ui.control.Button
        MeteotsunamiLabel               matlab.ui.control.Label
        Label_11                        matlab.ui.control.Label
        Tab_3                           matlab.ui.container.Tab
        AddrollereffectsCheckBox        matlab.ui.control.CheckBox
        TimeStepSedToMorphlEditField    matlab.ui.control.NumericEditField
        TimeStepSedToMorpEditFieldLabel  matlab.ui.control.Label
        ButtonGroup_21                  matlab.ui.container.ButtonGroup
        Button_37                       matlab.ui.control.RadioButton
        SedimentbedButton               matlab.ui.control.RadioButton
        HardbednoerosionButton          matlab.ui.control.RadioButton
        BottomTypeLabel                 matlab.ui.control.Label
        Button_35                       matlab.ui.control.Button
        FileEditField                   matlab.ui.control.EditField
        FileEditFieldLabel              matlab.ui.control.Label
        RungeKuttaparam2EditField_2     matlab.ui.control.NumericEditField
        RungeKuttaparam2EditFieldLabel_2  matlab.ui.control.Label
        ButtonGroup_22                  matlab.ui.container.ButtonGroup
        Button_38                       matlab.ui.control.RadioButton
        NoButton_7                      matlab.ui.control.RadioButton
        YesButton_8                     matlab.ui.control.RadioButton
        ConsideravalancheLabel          matlab.ui.control.Label
        MorphologicalChangeLabel_2      matlab.ui.control.Label
        AvalancheLabel                  matlab.ui.control.Label
        TimeStepBouss                   matlab.ui.control.NumericEditField
        EditFieldLabel                  matlab.ui.control.Label
        BedChangeLabel                  matlab.ui.control.Label
        ButtonGroup_20                  matlab.ui.container.ButtonGroup
        Button_36                       matlab.ui.control.RadioButton
        NobedchangeButton               matlab.ui.control.RadioButton
        UpdatedepthButton               matlab.ui.control.RadioButton
        MinDepthSedPickUpEditField      matlab.ui.control.NumericEditField
        MinDepthSedPickUpEditFieldLabel  matlab.ui.control.Label
        RungeKuttaparameter2EditField   matlab.ui.control.NumericEditField
        RungeKuttaparameter2EditFieldLabel  matlab.ui.control.Label
        RungeKuttaparameter1EditField   matlab.ui.control.NumericEditField
        RungeKuttaparameter1EditFieldLabel  matlab.ui.control.Label
        ShieldsparamBedloadEditField    matlab.ui.control.NumericEditField
        ShieldsparameterbedloadEditFieldLabel  matlab.ui.control.Label
        CriticalShieldsEditField        matlab.ui.control.NumericEditField
        CriticalShieldsEditFieldLabel   matlab.ui.control.Label
        SettlingvelocitymsEditField     matlab.ui.control.NumericEditField
        SettlingvelocitymsEditFieldLabel  matlab.ui.control.Label
        SedimentporosityEditField       matlab.ui.control.NumericEditField
        SedimentporosityEditFieldLabel  matlab.ui.control.Label
        SedimentdensityEditField        matlab.ui.control.NumericEditField
        SedimentdensityEditFieldLabel   matlab.ui.control.Label
        MediangrainDMEditField          matlab.ui.control.NumericEditField
        MediangrainDMEditFieldLabel     matlab.ui.control.Label
        NumericalschemeLabel            matlab.ui.control.Label
        ButtonGroup_19                  matlab.ui.container.ButtonGroup
        TVDButton                       matlab.ui.control.RadioButton
        UpwindingButton                 matlab.ui.control.RadioButton
        SedimentLabel                   matlab.ui.control.Label
        Label_12                        matlab.ui.control.Label
        Label_5                         matlab.ui.control.Label
        Label_6                         matlab.ui.control.Label
        Tab_6                           matlab.ui.container.Tab
        SelectMethodLabel               matlab.ui.control.Label
        ValueEditField                  matlab.ui.control.NumericEditField
        ValueEditFieldLabel_2           matlab.ui.control.Label
        FrictionCoefficientEditField    matlab.ui.control.NumericEditField
        FrictionCoefficientEditFieldLabel  matlab.ui.control.Label
        ViscosityCheckBox               matlab.ui.control.CheckBox
        FrictionCheckBox_2              matlab.ui.control.CheckBox
        ShockcapturingCheckBox          matlab.ui.control.CheckBox
        ActivateCheckBox                matlab.ui.control.CheckBox
        IncludeSedimentEffectintheModelCheckBox  matlab.ui.control.CheckBox
        MinimumClearanceEditField       matlab.ui.control.NumericEditField
        MinimumClearanceLabel           matlab.ui.control.Label
        DeepDraftVesselsLabel           matlab.ui.control.Label
        Label_8                         matlab.ui.control.Label
        VesselLabel                     matlab.ui.control.Label
        Button_40                       matlab.ui.control.Button
        ImportvesselfilesTextArea       matlab.ui.control.TextArea
        ImportVesselFilesLabel          matlab.ui.control.Label
        Label_13                        matlab.ui.control.Label
        ExpectedOutputTab               matlab.ui.container.Tab
        GridLayout                      matlab.ui.container.GridLayout
        ThresholdEditField              matlab.ui.control.NumericEditField
        ThresholdEditFieldLabel         matlab.ui.control.Label
        MaximumWaveHeightCheckBox       matlab.ui.control.CheckBox
        MinimumWaveHeightCheckBox       matlab.ui.control.CheckBox
        VVelocitymeanCheckBox           matlab.ui.control.CheckBox
        UVelocitymeanCheckBox           matlab.ui.control.CheckBox
        GenerateinputtxtButton_2        matlab.ui.control.Button
        OthersLabel                     matlab.ui.control.Label
        TimeLabel                       matlab.ui.control.Label
        FluxLabel                       matlab.ui.control.Label
        MaskLabel                       matlab.ui.control.Label
        SourceLabel                     matlab.ui.control.Label
        WaveHeightLabel                 matlab.ui.control.Label
        VelocityLabel                   matlab.ui.control.Label
        UndertowCheckBox                matlab.ui.control.CheckBox
        MaximumVelocityCheckBox         matlab.ui.control.CheckBox
        BreakingAgeCheckBox             matlab.ui.control.CheckBox
        GyFluxCheckBox                  matlab.ui.control.CheckBox
        GxFluxCheckBox                  matlab.ui.control.CheckBox
        FyFluxCheckBox                  matlab.ui.control.CheckBox
        FxFluxCheckBox                  matlab.ui.control.CheckBox
        QmomentumFluxCheckBox           matlab.ui.control.CheckBox
        PMomentumFluxCheckBox           matlab.ui.control.CheckBox
        BreakingLocationCheckBox        matlab.ui.control.CheckBox
        YSourceCheckBox                 matlab.ui.control.CheckBox
        PressureFieldCheckBox           matlab.ui.control.CheckBox
        XSourceCheckBox                 matlab.ui.control.CheckBox
        RollerinducedFluxCheckBox       matlab.ui.control.CheckBox
        WetdrymaskforBoussinesqNSWECheckBox  matlab.ui.control.CheckBox
        WaveHeightCheckBox              matlab.ui.control.CheckBox
        WetdrymaskCheckBox              matlab.ui.control.CheckBox
        SurfaceElevationattimetCheckBox  matlab.ui.control.CheckBox
        ArrivalCheckBox                 matlab.ui.control.CheckBox
        VVelocityCheckBox               matlab.ui.control.CheckBox
        MaxMomentumFluxCheckBox         matlab.ui.control.CheckBox
        UVelocityCheckBox               matlab.ui.control.CheckBox
        MaxVorticityCheckBox            matlab.ui.control.CheckBox
        DepthCheckBox                   matlab.ui.control.CheckBox
        TabGroup3                       matlab.ui.container.TabGroup
        LogReportTab                    matlab.ui.container.Tab
        Preview_LogReport               matlab.ui.control.TextArea
        inputtxtTab                     matlab.ui.container.Tab
        Preview_Input                   matlab.ui.control.TextArea
        Makefile                        matlab.ui.container.Tab
        Preview_Makefile                matlab.ui.control.TextArea
    end


    properties (Access = private)
        filename
        path
        inputpath
        inputdefault
        outputfile
        ouputdirectoryfinal
        SWE
        DspongeSwitch
        DspongeDecay
        initialHeight
        initialXvelocity
        initialYvelocity
        Mglob
        Nglob
        NumberStations
        STATION_FILE
        bathymetryinputdata
        NestStation
```

401	WORKFOLDER
402	GAUGEFILE
403	DEPTHFILE
404	GaugeListGrids
405	GaugeFname
406	Cd_file
407	filenamefriction
408	SurfaceHeightNAME
409	XVelocityNAME
410	YVelocityNAME
411	SurfaceHeightLOC
412	XVelocityLOC
413	YVelocityLOC
414	obstacleLOC
415	obstacleNAME
416	WaveFileNAME
417	WaveFileLOC
418	wavemaker
419	ConsWindNAME
420	ConsWindLOC
421	StormFileNAME
422	StormFileLOC
423	PressureFileNAME
424	PressureFileLOC
425	HardBottomFileNAME
426	HardBottomFileLOC
427	VesselNumber
428	VesselLOC
429	VesselNAME
430	HardbottomLOC
431	HardbottomNAME
432	CouplingLOC
433	CouplingNAME
434	mpif90val
435	directory
436	logreport
437	MakefileContent
438	processPID
439	makefile
440	depthfilelocation
441	depthdata
442	depthtype
443	rasterdegrees
444	depthfinal
445	GAUGEFILEllocation
446	colsChildLayer
447	rowsChildLayer
448	directorylog
449	CouplingLOC2
450	fortranFilePath0
451	gfortranReport
452	resultCompile
453	resultRun
454	inputFolder
455	CouplingX
456	CouplingY
457	CoupX
458	CoupY

```
        GAUGEFILElocation
        latGauge
        gaugetext0
        CoupMatrixON
        gridX
        gridY
        row1
        col1
    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)
            workingfolder = uigetdir;
            workingfolder = fullfile(workingfolder);

            %Create the 'INPUT_FILES' folder in the working directory
            app.WORKFOLDER = fullfile(string(workingfolder), 'INPUT_FILES');
            app.SavefilestoEditField.Value = fullfile(string(workingfolder), 'INPUT_FILES');
            app.SavefilestoEditField.FontColor = 'k';
        end

        % Button pushed function: Button_8
        function Button_8Pushed(app, event)
            %Load the bathymetry file and display its filename in the textbox
            [bathymetryfile,path2] = uigetfile('*.txt;*.tif');
            app.depthfilelocation  = string(path2);
            app.bathymetryinputdata = string(path2) + string(bathymetryfile);
            app.DepthFileEditField.BackgroundColor = 'w';
            app.DepthFileEditField.FontColor =  'k';

%             try
                if contains(string(app.bathymetryinputdata),'.txt')
                    app.depthtype = 1;
                    A1 = readmatrix(app.bathymetryinputdata);
                    app.depthdata = A1;
                    app.DEPTHFILE = string(bathymetryfile);

                    %Extract M and N global dimensions
                    [row,col] = size(A1);
                    app.XEditField_2.Value = col;
                    app.YEditField_2.Value = row;
                    app.XEditField_2.FontColor = 'k';
                    app.YEditField_2.FontColor = 'k';
                    app.ButtonGroup_4.SelectedObject = app.Button_2;

                elseif contains(string(app.bathymetryinputdata),'.tif')
                    %Extract information from the bathymetry
                    app.depthtype = 2;
                    [A1,R2] = readgeoraster(app.bathymetryinputdata);
                    A =  size(A1);
                    app.depthdata = flipud(A1); %FUNWAVE reads the depth file starting from the southwest corner
                    %The readgeoraster function generates a matrix with the first point representing the northwest corner

                    bathname= erase(bathymetryfile, '.tif');  % Remove .tif extension
                    app.DEPTHFILE = [bathname, '.txt'];       % Append .txt extension


                    %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';
                    app.LatitudeEditField.Enable = "on";
                    app.LongitudeEditField.Enable = "on";

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

                    %Extract M and N global dimenstions
                    app.XEditField_2.Value = A(1,2);
                    app.YEditField_2.Value = A(1,1);
                    app.col1 = A(1,2);
                    app.row1 = A(1,1);
                    app.XEditField_2.FontColor = 'k';
                    app.YEditField_2.FontColor = 'k';

                    if R2.AngleUnits == "degrees"
                        app.ButtonGroup_4.SelectedObject = app.SphericalButton;
                        app.SphericalButton.FontColor = 'k';
                        app.CartesianButton.FontColor = 'k';
                    end
                    app.rasterdegrees = R2;
                else
                    app.ButtonGroup_4.SelectedObject = app.Button_2;
                    app.DepthFileEditField.Value = '';
                end
                    app.DepthFileEditField.Value = app.DEPTHFILE; %Display the filename in the textbox
%             end

        end

        % Callback function
        function TotalTimeEditFieldValueChanged(app, event)

        end

        % Callback function
        function PlotintervalEditFieldValueChanged(app, event)


        end

        % Button pushed function: GenerateinputtxtButton
        function GenerateinputtxtButtonPushed(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.



% 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:


            %--------------------------------------------
            %------Remove old Makefile, funwave.exe-----
            %--------------------------------------------
            %In reclicking the button, delete existing Makefile and funwave.exe
            if exist(fullfile(app.directory, "funwave"),'file')
                delete(fullfile(app.directory, "funwave"));
            end

            if exist(fullfile(app.directory,"Makefile"),'file')
                delete(fullfile(app.directory,"Makefile"));
            end

            %Disable the Makefile  buttons
            app.CreateMakefileButton.Enable = 'off';
            app.StartsimulationButton.Enable = 'off';

            %Remove preview in the Makefile Tab
            app.Preview_Makefile.Value = '';

            %----------------------------
            %------Output Directory------
            %----------------------------
            %Check if the output directory has been manually set by the user
            if  app.SavefilestoEditField.Value == string(app.WORKFOLDER)
                inputtxt_directory =  fullfile(app.WORKFOLDER);
                app.SavefilestoEditField.FontColor = 'k';

                %Check if the 'INPUT_FILES' directory exists, and create it if it doesn't
                if ~exist(inputtxt_directory, 'dir')
                    mkdir(inputtxt_directory); % Creates the directory
                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
                    %Handle other platforms
                    defaultDir = pwd;
                end

                    Dir1 = fullfile(defaultDir, 'INPUT_FILES'); %Create 'INPUT_FILES' folder

                    % Check if the 'INPUT_FILES' directory exists, and create it if it doesn't
                    if ~exist(Dir1, 'dir')
                        mkdir(Dir1); % Creates the directory
                    end

                    app.SavefilestoEditField.Value = deblank(string(Dir1));  %"Current Folder";
                    inputtxt_directory =  deblank(string(Dir1));
                    app.SavefilestoEditField.FontColor = 'k';
            end

            app.directory  = inputtxt_directory;


            %----------------------------------
            %-----CREATE INPUT.TXT HEADER------
            %----------------------------------
            %Create the input.txt
            fileInputPath = fullfile(inputtxt_directory, 'input.txt');
            fileInputId = fopen(fileInputPath, 'w');

            %Template of the first few lines of the input.txt
            templateStr = [
                '! INPUT FILE FOR BOUSS_TVD\n' ...
                '! Note: all input parameter are capital sensitive\n\n' ...
                '! Generated using FUNMAP by Felix, et al. (2025) \n\n'...
                '! --------------------TITLE-------------------------------------\n' ...
                'TITLE = FUNWAVE\n\n' ...
                '  ! -------------------HOT START---------------------------------\n' ...
                'HOT_START = F\n' ...
                'FileNumber_HOTSTART = HotStartMTLB\n'];

            %Add the template into the text file
            fprintf(fileInputId, templateStr);

            %Create Input_Data subfolder
            inputFolder = fullfile(inputtxt_directory, 'Input_Data');
            app.inputFolder = inputFolder;

            %Check if the 'INPUT_FILES/Input_Data' directory exists, create it if it doesn't
            if ~exist(app.inputFolder, 'dir')
                mkdir(app.inputFolder);  % Creates the 'INPUTFOLDER' directory
            end


            %----------------------
            %------LOG REPORT------
            %----------------------
            %Create 'Log_Files' subfolder
            LogFolder = fullfile(inputtxt_directory, 'Log_Files');
            LogFile = fullfile(LogFolder,'Log_Report_InputPrep.txt');

                % Check if the 'INPUT_FILES/Log_Files' directory exists, create it if it doesn't
                if ~exist(LogFolder, 'dir')
                    mkdir(LogFolder);
                end
```

```matlab

%Create the log text file
fileId = fopen(fullfile(LogFolder,'Log_Report_InputPrep.txt'),'w');

%Template for the default section of the log report
fprintf(fileId, '%s\n','******************** LOG REPORT ******************** ');
fprintf(fileId, '%-30s%-s\n', 'Type:', 'input.txt, Makefile, funwave executable');
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','');
fprintf(fileId,'%-s\n\n','*******  INPUT.TXT SECTION *******');

error = 0; % General error counter
error_mkfile = 0; %Makefile error counter

%-------------------------
%------PARALLEL INFO------
%-------------------------
tempParallel = sprintf(['\n! --------------------PARALLEL INFO-----------------------------\n' ...
    '!    PX,PY - processor numbers in X and Y\n' ...
    '!    NOTE: make sure consistency with mpirun -np n (px*py)\n' ...
    'PX = %.0f\n' ...
    'PY = %.0f\n'], ...
    app.ProcessorX.Value, ...
    app.ProcessorY.Value);

%Add these lines in the input.txt
fprintf(fileInputId, '%s', tempParallel);

%-----------------
%------DEPTH------
%-----------------
%If no option is selected
if app.Button4.Value
    app.ImportDataButton.FontColor = 'r';
    app.FlatButton.FontColor = 'r';
    app.SlopeButton.FontColor = 'r';
    fprintf(fileId,'%-30s%-s\n','DEPTH:', 'No data is selected');
    error = error + 1; %to note an error occurred
end

%------------------------------
%------DEPTH: Import Data------
%------------------------------
if app.ImportDataButton.Value
    % Check if no file is uploaded
    Depthconditions = [...
    isempty(app.DepthFileEditField.Value), ...
    ~strcmp(app.DepthFileEditField.Value, app.DEPTHFILE) ...
    str2double(app.DepthFileEditField.Value) == 0 ...
    ];

%Display 'NO FILE' in the textbox and note the issue in the log report
if any(Depthconditions)
    app.DepthFileEditField.BackgroundColor = 'r';
    app.DepthFileEditField.Value = 'NO FILE';
    app.DepthFileEditField.FontColor = 'w';
    fprintf(fileId,'%-30s%-s\n','DEPTH:', 'No file is uploaded');
    error = error + 1;

else %Read the bathymetry file
    if app.InvertvaluesCheckBox.Value
        %Convert water depth values to positive values
        depth = -app.depthdata;
    else
        depth = app.depthdata;
    end

    % Round off depth to the nearest five decimal places
    if isnumeric(depth) && isscalar(depth)
        %Scalar numeric input
        app.depthfinal = round(depth, 5);

    elseif ischar(depth) || isstring(depth)
        %String input
        depthNum = str2double(depth);
        if ~isnan(depthNum)
            app.depthfinal = round(depthNum, 5);
        end
    elseif isnumeric(depth) && ~isscalar(depth)
        %Numeric array inpput
        app.depthfinal = round(double(depth), 5);
    else
        app.depthfinal = round(depth, 5);
    end

    %Save the bathymetry file in 'INPUT_FILES/Input_Data/'
    outputFilePath = fullfile(app.inputFolder,app.DEPTHFILE);


    % Open the file for writing
    fileID0 = fopen(outputFilePath, 'w');

    % Define the format for each row (e.g., '%.5f' for two decimal places)
    formatSpec = [repmat('%.5f\t', 1, size(app.depthfinal, 2) - 1), '%.5f\n'];

    % Write each row of depthfinal to the file
    for row = 1:size(app.depthfinal, 1)
        fprintf(fileID0, formatSpec, app.depthfinal(row,
```

```matlab
:));
                    end

                    % Close the file
                    fclose(fileID0);



                    tempDepth = sprintf([
                        '\n  ! ---------------------DEPTH-------------------------------------\n' ...
                        'DEPTH_TYPE = DATA\n' ...
                        'DEPTH_FILE =  Input_Data/%s\n' ...
                        'DepthFormat = ELE\n' ...
                        'WaterLevel = %.1f\n'], ...
                        app.DEPTHFILE,app.WaterlevelmEditField.Value);
                end
            end

            %-----------------------
            %------DEPTH: Flat------
            %-----------------------
            if app.FlatButton.Value
                tempDepth = sprintf([
                    '\n  ! ---------------------DEPTH-------------------------------------\n' ...
                    'DEPTH_TYPE = FLAT\n' ...
                    'DEPTH_FLAT = %.2f\n'], ...
                    app.DepthmEditField.Value);

                % Add warning note in the log report if the 0 value is unchanged
                if app.DepthmEditField.Value == 0
                    app.DepthmEditField.FontColor = 'r';
                    fprintf(fileId,'%-30s%-s\n','DEPTH:', 'Flat depth is zero');
                    error = error + 1;
                end
            end

            %------------------------
            %------DEPTH: Slope------
            %------------------------
            if app.SlopeButton.Value
                tempDepth = sprintf([
                    '\n  ! ---------------------DEPTH-------------------------------------\n' ...
                    'DEPTH_TYPE = SLOPE\n' ...
                    'DEPTH_FLAT = %.2f\n' ...
                    'SLP = %.2f\n' ...
                    'Xslp = %.2f\n'], ...
                    app.DepthmEditField.Value, ...
                    app.AngleEditField.Value, ...
                    app.StartatXEditField.Value);


                %If one of the depth options is selcted
                if exist('tempDepth', 'var') == 1
                    fprintf(fileInputId, '%s', tempDepth); %Add the Depth info in the input.txt
                end

                % Add warning note in the log report if 0 value is unchanged
                if app.DepthmEditField.Value == 0
                    app.DepthmEditField.FontColor = 'r';
                    fprintf(fileId,'%-30s%-s\n','DEPTH:', 'Flat depth is zero');
                    error = error + 1;
                end

                if app.DepthmEditField.Value == 0
                    app.DepthmEditField.FontColor = 'r';
                    fprintf(fileId,'%-30s%-s\n','DEPTH:', 'Slope depth is zero');
                    error = error + 1;
                end

                if app.AngleEditField.Value == 0
                    app.AngleEditField.FontColor = 'r';
                    fprintf(fileId,'%-30s%-s\n','DEPTH:', 'Slope angle is zero');
                    error = error + 1;
                end

            end

            if exist("tempDepth",'var')
                fprintf(fileInputId, '%s', tempDepth);
            end

            %---------------------
            %------DIMENSION------
            %---------------------
            tempDim = sprintf([
                '\n  ! ------------------DIMENSION-----------------------------\n' ...
                '  ! global grid dimension\n' ...
                'Mglob = %d\n' ...
                'Nglob = %d\n'], ...
                app.XEditField_2.Value, ...
                app.YEditField_2.Value);

            %Add these lines in the input.txt
            fprintf(fileInputId, '%s', tempDim);

            %Add warning note in the log report if the 0 value is unchanged
            if app.XEditField_2.Value == 0
                app.XEditField_2.FontColor = 'r';
                fprintf(fileId,'%-30s%-s\n','DIMENSION:', 'X is zero');
                error = error + 1;
            end

            if app.YEditField_2.Value == 0
                app.YEditField_2.FontColor = 'r';
```

```
                    fprintf(fileId,'%-30s%-s\n','DIMENSION:', 'Y is zero');
                    error = error + 1;
                end

                %-----------------
                %------PRINT------
                %-----------------
                % Identify the correct directory separator based on the operating system
                if ismac
                    dirSeparator = "/";
                elseif ispc
                    dirSeparator = "\";
                else
                    dirSeparator = "/";
                end

                if isempty(app.ResultsfolderEditField.Value)
                    %Use 'Simulation_Results' as the default folder name
                    RESULT_FOLDER = "Simulation_Results" + dirSeparator;
                    app.ResultsfolderEditField.Value = RESULT_FOLDER;
                    app.ResultsfolderEditField.FontColor = 'k';
                else %Use the user's input
                    % if endsWith(app.ResultsfolderEditField.Value, dirSeparator)
                        RESULT_FOLDER = app.ResultsfolderEditField.Value;
                    % else
                    %     RESULT_FOLDER = app.ResultsfolderEditField.Value + dirSeparator;
                    %     app.ResultsfolderEditField.Value = RESULT_FOLDER;
                    % end
                end

                tempPrint = sprintf([
                    '\n  ! -------------------PRINT---------------------------------\n' ...
                    'RESULT_FOLDER = %s\n'], ...
                    RESULT_FOLDER); %

                %Add these lines in the input.txt
                fprintf(fileInputId, '%s', tempPrint);

                %----------------
                %------TIME------
                %----------------
                tempTime = sprintf([
                    '\n  ! -----------------TIME----------------------------------\n' ...
                    '  ! time: total computational time / plot time / screen interval \n' ...
                    '  ! all in seconds\n' ...
                    'PLOT_START_TIME = %.1f\n' ...
                    'TOTAL_TIME = %.1f\n' ...
                    'PLOT_INTV = %.1f\n' ...
                    'PLOT_INTV_STATION = %.1f\n' ...
                    'SCREEN_INTV = %.1f\n'], ...
                    app.StartEditField.Value, ...
                    app.TotalEditField.Value, ...
```

```
    app.SimulationIntervalEditField.Value, ...
    app.GaugeIntervalEditField.Value, ...
    app.SimulationIntervalEditField.Value);

%Add these lines in the input.txt
fprintf(fileInputId, '%s', tempTime);

%----------------
%------GRID------
%----------------
tempGrid = sprintf([
    '\n  ! -----------------GRID----------------------------------\n' ...
    'StretchGrid = F\n']);

fprintf(fileInputId, '%s', tempGrid);

%---------------------------
%------GRID: Spherical------
%---------------------------
%Changing the parameter variable for Grid Size based on the coordinate system
if app.SphericalButton.Value
    tempGrid2 = sprintf([
        '!spherical grid, in decimal degrees\n' ...
        'Lon_West = %.4f\n' ...
        'Lat_South = %.4f\n' ...
        'Dphi = %.4f\n' ...
        'Dtheta = %.4f\n'], ...
        app.LongitudeEditField.Value, app.LatitudeEditField.Value, app.XEditField.Value, app.YEditField.Value);

    %Add warning note in the log report if the 0 value is unchanged
    if app.LongitudeEditField.Value == 0
        app.LongitudeEditField.FontColor = 'r';
        fprintf(fileId,'%-30s%-s\n','GRID:', 'Longitude is zero');

        error = error + 1;
    end
    if app.LatitudeEditField.Value == 0
        app.LatitudeEditField.FontColor = 'r';
        fprintf(fileId,'%-30s%-s\n','GRID:', 'Latitude is zero');

        error = error + 1;
    end

    if app.LatitudeEditField.Value == 0 && app.LongitudeEditField.Value == 0
        error_mkfile = error_mkfile +1;
        error_mkfiletext3 = sprintf('Grid');
    end

    if app.XEditField.Value == 0
        app.XEditField.FontColor = 'r';
        fprintf(fileId,'%-30s%-s\n','GRID:', 'X is zero');
        error = error + 1;
    end
```

```
                    if app.YEditField.Value == 0
                        app.YEditField.FontColor = 'r';
                        fprintf(fileId,'%-30s%-s\n','GRID:', 'Y is zero');
                        error = error + 1;
                    end

                    if app.XEditField.Value == 0 && app.YEditField.Value == 0
                        error_mkfile = error_mkfile +1;
                        error_mkfiletext3 = sprintf('Grid');
                    end

                    if app.ArrivalCheckBox.Value
                        % Include the arrival time threshold
                        tempGrid2 = [tempGrid2, sprintf('ArrTimeMinH = %.4f\n', app.ThresholdEditField.Value)];
                    end

                %----------------------------
                %------GRID: Cartesian------
                %----------------------------
                elseif app.CartesianButton.Value
                    tempGrid2 = sprintf([
                        '! cartesian grid sizes\n' ...
                        'DX = %.2f\n' ...
                        'DY = %.2f\n'], ...
                        app.XEditField.Value, app.YEditField.Value);

                    app.LatitudeEditField.FontColor = '0.65,0.65,0.65';
                    app.LongitudeEditField.FontColor = '0.65,0.65,0.65';

                    if app.XEditField.Value == 0
                        app.XEditField.FontColor = 'r';
                        fprintf(fileId,'%-30s%-s\n','GRID:', 'X is zero');
                        error = error + 1;
                    end
                    if app.YEditField.Value == 0
                        app.YEditField.FontColor = 'r';
                        fprintf(fileId,'%-30s%-s\n','GRID:', 'Y is zero');
                        error = error + 1;
                    end

                    if app.XEditField.Value == 0 && app.YEditField.Value == 0
                        error_mkfile = error_mkfile +1;
                        error_mkfiletext3 = sprintf('Grid');
                    end

                %-----------------------
                %------GRID: None------
                %-----------------------
                elseif app.Button_2.Value
                    %If no Cooordinate system is selected, add an error in the log report
                    app.SphericalButton.FontColor = 'r';
                    app.CartesianButton.FontColor = 'r';
                    fprintf(fileId,'%-30s%-s\n','GRID:', 'No coordinate system is selected ');
                    tempGrid2 = '';
                    error = error + 1;
                    error_mkfile = error_mkfile +1;
                    error_mkfiletext3 = sprintf('Grid');
                end

                %Add these lines in the input.txt
                fprintf(fileInputId, '%s', tempGrid2);

                %-----------------------------
                %------INITIAL CONDITION------
                %-----------------------------
                if app.Button_31.Value
                    fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:', 'None↙
    selected. Set it up in Tab 2');
                    error = error + 1;
                end

                %-----------------------------------------
                %------INITIAL CONDITION: Import UVZ------
                %-----------------------------------------
                if app.ImportUVZButton.Value
                    tempUVZ1 = sprintf([
                        '\n  ! --------------- INITIAL UVZ↙
    ---------------------------\n' ...
                        '  ! INI_UVZ - initial UVZ e.g., initial↙
    deformation\n' ...
                        '  !         must provide three (3) files \n' ...
                        'INI_UVZ = T\n']);

                    %Set up the initial values for the three parameters to↙
    avoid errors when creating the text file
                    tempUVZ2 = sprintf('ETA_FILE = \n');
                    tempUVZ3 = sprintf('U_FILE = \n');
                    tempUVZ4 = sprintf('V_FILE = \n');

                    %Check if the eta file / Z file is loaded
                    ETAconditions = [...
                        isempty(app.SurfaceHeightEditField.Value), ...
                        ~any(strcmp(app.SurfaceHeightEditField.Value, app.↙
    SurfaceHeightNAME)), ...
                        str2double(app.SurfaceHeightEditField.Value) == 0 ...
                        ];

                    if any(ETAconditions)
                        %Display 'NO FILE' in the textbox and note the issue↙
    in the log report
                        fprintf(fileId, '%-30s%-s\n', 'INITIAL CONDITION:',↙
    'Import UVZ:');
                        fprintf(fileId, '%-30s%-s\n', '', 'No file uploaded in↙
    ''Surface Height''');
                        app.SurfaceHeightEditField.Value = "NO FILE";
                        app.SurfaceHeightEditField.FontColor = 'w';
                        app.SurfaceHeightEditField.BackgroundColor = 'r';
                        error = error + 1;
                    elseif strcmp(app.SurfaceHeightEditField.Value, app.↙
    SurfaceHeightNAME)
                       %If the filename matches the name in the textbox, add↙
    it to input.txt
                        tempUVZ2 = sprintf('ETA_FILE = Input_Data/%s\n', app.↙
            SurfaceHeightNAME);

                        %If the file is not in 'INPUT_FILES/Input_Data', copy it there
                        if ~exist(fullfile(app.inputFolder, app.SurfaceHeightNAME), 'file')
                            copyfile(app.SurfaceHeightLOC, app.inputFolder);
                        end
                    end

                    % Checking if U vector file is loaded
                    uConditions = [...
                        isempty(app.UVelocityEditField.Value), ... % Check if the field is empty
                        ~any(strcmp(app.UVelocityEditField.Value, app.XVelocityNAME)), ... % Check if the value does not match expected names
                        str2double(app.UVelocityEditField.Value) == 0 ... % Check if the value is zero (assuming numeric comparison is relevant)
                        ];

                    if any(uConditions)
                        %Display 'NO FILE' in the textbox and note the issue in the log report
                        fprintf(fileId, '%-30s%-s\n', 'INITIAL CONDITION:', 'Import UVZ:');
                        fprintf(fileId, '%-30s%-s\n', '', 'No file uploaded in ''U Velocity''');
                        app.UVelocityEditField.Value = "NO FILE";
                        app.UVelocityEditField.FontColor = 'w';
                        app.UVelocityEditField.BackgroundColor = 'r';
                        error = error + 1;
                    else
                        %If the filename matches the name in the textbox, add it to input.txt
                        tempUVZ3 = sprintf('U_FILE = Input_Data/%s\n', app.XVelocityNAME);

                        %If the file is not in 'INPUT_FILES/Input_Data', copy it there
                        if ~exist(fullfile(app.inputFolder, app.XVelocityNAME), 'file')
                            copyfile(app.XVelocityLOC, app.inputFolder);
                        end
                    end

                    % Checking if V vector file is loaded
                    vConditions = [...
                        isempty(app.VVelocityEditField.Value), ... % Check if the field is empty
                        ~any(strcmp(app.VVelocityEditField.Value, app.YVelocityNAME)), ... % Check if the value does not match expected names
                        str2double(app.VVelocityEditField.Value) == 0 ... % Check if the value is zero (assuming numeric comparison is relevant)
                        ];

                    if any(vConditions)
                        %Display 'NO FILE' in the textbox and note the issue in the log report
```

```
                        fprintf(fileId, '%-30s%-s\n', 'INITIAL CONDITION:', 'Import UVZ:');
                        fprintf(fileId, '%-30s%-s\n', '', 'No file uploaded in ''V Velocity''');
                        app.VVelocityEditField.Value = "NO FILE";
                        app.VVelocityEditField.FontColor = 'w';
                        app.VVelocityEditField.BackgroundColor = 'r';
                        error = error + 1;
                    else
                        %If the filename matches the name in the textbox, add it to input.txt
                        tempUVZ4 = sprintf('V_FILE = Input_Data/%s\n', app.YVelocityNAME);

                        %If the file is not in 'INPUT_FILES/Input_Data', copy it there
                        if exist(fullfile(inputFolder, app.YVelocityNAME), 'file') == 2
                            %do nothing
                        else
                            %Copy the file to the OUTPUT_FOLDER
                            copyfile(app.YVelocityLOC, inputFolder);
                        end
                    end

                    % Combine all the the parts
                    tempUVZ = [tempUVZ1,tempUVZ2,tempUVZ3,tempUVZ4];

                    %Add these lines in the input.txt
                    fprintf(fileInputId, '%s', tempUVZ);
                else
                    fprintf(fileInputId, [
                        '\n  ! --------------- INITIAL UVZ ---------------------------\n' ...
                        'INI_UVZ = F\n']);
                end

                %----------------------------------------
                %------INITIAL CONDITION: Wavemaker------
                %----------------------------------------
                %To list all the errors in the log report
            function checkAndReport(field, fieldName, fileId)
                if field.Value == 0
                    field.FontColor = 'r';
                    fprintf(fileId, '%-30s%-s\n', 'INITIAL CONDITION:', 'Wavemaker:');
                    fprintf(fileId, '%-35s%-s\n', '', [fieldName ' is zero']);
                    error = error + 1;
                else
                    field.FontColor = 'k';
                end
            end

            if app.WavemakerButton.Value
                %Define variables
                Xc = app.XcoordinatemEditField.Value;
                Yc = app.YcoordinatemEditField.Value;
                WID = app.WidthmEditField.Value;
                AMP = app.AmplitudemEditField.Value;
                DEP = app.WaterdepthmEditField.Value;
                LAGTIME = app.LagtimesecEditField.Value;
                Tperiod = app.PeriodsecEditField.Value;
                Theta_WK = app.ThetadegreesEditField.Value;
                Time_ramp = app.TimerampsecEditField.Value;
                NumWaveComp = app.WaveCompCountEditField.Value;
                FreqPeak =  app.PeakEditField.Value;
                FreqMin = app.MinimumEditField.Value;
                FreqMax = app.MaximumEditField.Value;
                Hmo = app.WaveHeightmEditField.Value;


                tempWavemaker1 = sprintf(['\n! -----------------WAVEMAKER------------------------------\n']);

                %Add only the necessary parameters based on the dropdown selection
                switch app.WavemakertypeDropDown.Value

                    case "INI_REC"
                        %--------------------
                        %......INI_REC......
                        %--------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = INI_REC \nXc = %.4f\nYc = %.4f\nWID = %.2f\n', Xc, Yc, WID);

                        %Log report
                        checkAndReport(app.XcoordinatemEditField, 'X Coordinate', fileId);
                        checkAndReport(app.YcoordinatemEditField, 'Y Coordinate', fileId);
                        checkAndReport(app.WidthmEditField, 'Width (m)', fileId);


                    case "JON_1D"
                        %-------------------
                        %......JON_1D......
                        %-------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = JON_1D\nXc_WK = %.4f\nYc_WK = %.4f\nYwidth_WK = %.2f\nDEP_WK = %.2f\nTime_ramp = %.2f\nDelta_WK = %.2f\nFreqPeak = %.2f\nFreqMin = %.2f\nFreqMax = %.2f\nHmo = %.2f\nGammaTMA = 3.3\nNfreq = 45.0\n', Xc, Yc, WID,DEP, Time_ramp,app.WidthDeltaEditField.Value, FreqPeak,FreqMin,FreqMax,Hmo);

                        %Log report
                        checkAndReport(app.XcoordinatemEditField, 'X Coordinate', fileId);
                        checkAndReport(app.YcoordinatemEditField, 'Y Coordinate', fileId);
                        checkAndReport(app.WidthmEditField, 'Width (m)', fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth (m)', fileId);
                        checkAndReport(app.TimerampsecEditField, 'Time ramp (sec)', fileId);
                        checkAndReport(app.WidthDeltaEditField, 'Width Delta',
fileId);
                        checkAndReport(app.MinimumEditField, 'Frequency↙
Minimum', fileId);
                        checkAndReport(app.MaximumEditField, 'Frequency↙
Maximum', fileId);
                        checkAndReport(app.PeakEditField, 'Frequency Peak',↙
fileId);
                        checkAndReport(app.WaveHeightmEditField, 'Wave height↙
(hmo)', fileId);

                    case "JON_2D"
                        %-------------------
                        %......JON_2D......
                        %-------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = JON_2D\nXc_WK =↙
%.4f\nYc_WK = %.4f\nYwidth_WK = %.2f\nDEP_WK = %.2f\nTime_ramp = %.2f\nDelta_WK =↙
%.2f\nFreqPeak = %.2f\nFreqMin = %.2f\nFreqMax = %.2f\nHmo = %.2f\nGammaTMA = 3.3↙
\nThetaPeak = 0.0\nNfreq = 45.0\nNtheta = 24.0\n', Xc, Yc, WID,DEP, Time_ramp,app.↙
WidthDeltaEditField.Value, FreqPeak,FreqMin,FreqMax,Hmo);

                        %Log report
                        checkAndReport(app.XcoordinatemEditField, 'X↙
Coordinate', fileId);
                        checkAndReport(app.YcoordinatemEditField, 'Y↙
Coordinate', fileId);
                        checkAndReport(app.WidthmEditField, 'Width (m)',↙
fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth↙
(m)', fileId);
                        checkAndReport(app.TimerampsecEditField, 'Time ramp↙
(sec)', fileId);
                        checkAndReport(app.WidthDeltaEditField, 'Width Delta',↙
fileId);
                        checkAndReport(app.MinimumEditField, 'Frequency↙
Minimum', fileId);
                        checkAndReport(app.MaximumEditField, 'Frequency↙
Maximum', fileId);
                        checkAndReport(app.PeakEditField, 'Frequency Peak',↙
fileId);
                        checkAndReport(app.WaveHeightmEditField, 'Wave height↙
(hmo)', fileId);

                    case "TMA_1D"
                        %-------------------
                        %......TMA_1D......
                        %-------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = TMA_1D\nXc_WK =↙
%.4f\nYc_WK = %.4f\nYwidth_WK = %.2f\nDEP_WK = %.2f\nTime_ramp = %.2f\nDelta_WK =↙
%.2f\nFreqPeak = %.2f\nFreqMin = %.2f\nFreqMax = %.2f\nHmo = %.2f\nGammaTMA = 3.3↙
\nNfreq = 45.0\n', Xc, Yc, WID,DEP, Time_ramp,app.WidthDeltaEditField.Value,↙
FreqPeak,FreqMin,FreqMax,Hmo);

                        %Log report
                        checkAndReport(app.XcoordinatemEditField, 'X↙
Coordinate', fileId);
                        checkAndReport(app.YcoordinatemEditField, 'Y↙
Coordinate', fileId);
                        checkAndReport(app.WidthmEditField, 'Width (m)',↙
fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth (m)', fileId);
                        checkAndReport(app.TimerampsecEditField, 'Time ramp (sec)', fileId);
                        checkAndReport(app.WidthDeltaEditField, 'Width Delta', fileId);
                        checkAndReport(app.MinimumEditField, 'Frequency Minimum', fileId);
                        checkAndReport(app.MaximumEditField, 'Frequency Maximum', fileId);
                        checkAndReport(app.PeakEditField, 'Frequency Peak', fileId);
                        checkAndReport(app.WaveHeightmEditField, 'Wave height (hmo)', fileId);

                    case "LEF_SOL"
                        %-------------------
                        %......LEF_SOL......
                        %-------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = LEF_SOL\nAMP = %.2f\nDEP = %.2f\nLAGTIME = %.2f\n', AMP, DEP, LAGTIME);

                        %Log report
                        checkAndReport(app.AmplitudemEditField, 'Amplitude (m)', fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth (m)', fileId);
                        checkAndReport(app.LagtimesecEditField, 'Lag time (sec)', fileId);

                    case "INI_SOL"
                        %-------------------
                        %......INI_SOL......
                        %-------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = INI_SOL\nAMP = %.2f\nDEP = %.2f\nXWAVEMAKER = %.4f\n', AMP, DEP, Xc);

                        %Log report
                        checkAndReport(app.AmplitudemEditField, 'Amplitude (m)', fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth (m)', fileId);
                        checkAndReport(app.XcoordinatemEditField, 'X Coordinate', fileId);

                    case "WK_IRR"
                        %------------------
                        %......WK_IRR......
                        %------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = WK_IRR\nXc_WK = %.4f\nYc_WK = %.4f\nYwidth_WK = %.2f\nDEP_WK = %.2f\nTime_ramp = %.2f\nDelta_WK = %.2f\nFreqPeak = %.2f\nFreqMin = %.2f\nFreqMax = %.2f\nHmo = %.2f\nGammaTMA = 3.3\nThetaPeak = 0.0\nNfreq = 45.0\nNtheta = 24.0\nEqualEnergy = TRUE', Xc, Yc, WID, DEP, Time_ramp,app.WidthDeltaEditField.Value, FreqPeak,FreqMin,FreqMax,Hmo);

                        %Log report
                        checkAndReport(app.XcoordinatemEditField, 'X
    Coordinate', fileId);
                        checkAndReport(app.YcoordinatemEditField, 'Y Coordinate', fileId);
                        checkAndReport(app.WidthmEditField, 'Width (m)', fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth (m)', fileId);
                        checkAndReport(app.TimerampsecEditField, 'Time ramp (sec)', fileId);
                        checkAndReport(app.WidthDeltaEditField, 'Width Delta', fileId);
                        checkAndReport(app.MinimumEditField, 'Frequency Minimum', fileId);
                        checkAndReport(app.MaximumEditField, 'Frequency Maximum', fileId);
                        checkAndReport(app.PeakEditField, 'Frequency Peak', fileId);
                        checkAndReport(app.WaveHeightmEditField, 'Wave height (hmo)', fileId);

                    case "WK_REG"
                        %------------------
                        %......WK_REG......
                        %------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = WK_REG\nXc_WK = %.4f\nYc_WK = %.4f\nYwidth_WK = %.2f\nTperiod = %.2f\nAMP_WK = %.2f\nDEP_WK = %.2f\nTheta_WK = %.2f\nTime_ramp = %.2f\nDelta_WK = %.2f\n!The Delta_WK option is activated as it is included in the simple_cases/sediment_rip file\n', Xc, Yc, WID, Tperiod, AMP, DEP, Theta_WK, Time_ramp,app.WidthDeltaEditField.Value);

                        %Log report
                        checkAndReport(app.XcoordinatemEditField, 'X Coordinate', fileId);
                        checkAndReport(app.PeriodsecEditField, 'Period (sec)', fileId);
                        checkAndReport(app.AmplitudemEditField, 'Amplitude (m)', fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth (m)', fileId);
                        checkAndReport(app.ThetadegreesEditField, 'Theta (degrees)', fileId);
                        checkAndReport(app.TimerampsecEditField, 'Time ramp (sec)', fileId);

                    case "WK_TIME_SERIES"
                        %-------------------------
                        %......WK_TIME_SERIES......
                        %-------------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = WK_TIME_SERIES\nWaveCompFile = Input_Data/%s\n NumWaveComp = %.2f\nPeakPeriod = %.2f\nDEP_WK = %.2f\nXc_WK = %.4f\nYwidth_WK = %.4f\n', app.WavecomponentfileEditField.Value, NumWaveComp, Tperiod, DEP, Xc, WID);

                        checkAndReport(app.WaveCompCountEditField, 'Wave component count', fileId);
                        checkAndReport(app.PeriodsecEditField, 'Period (sec)', fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth
    (m)', fileId);
                        checkAndReport(app.XcoordinatemEditField, 'X Coordinate', fileId);
                        checkAndReport(app.WidthmEditField, 'Width (m)', fileId);

                        %Check if the file is loaded
                        wavecompconditions = [...
                            isempty(app.WavecomponentfileEditField.Value ), ... % Check if the field is empty
                            ~any(strcmp(app.WavecomponentfileEditField.Value , app.WaveFileNAME)), ... % Check if the value does not match expected names
                            str2double(app.WavecomponentfileEditField.Value ) == 0 ... % Check if the value is zero (assuming numeric comparison is relevant)
                            ];

                        if any(wavecompconditions)
                            %Display 'NO FILE' in the textbox and note the issue in the log report
                            fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:','Wavemaker:');
                            fprintf(fileId, '%-35s%-s\n', '', 'No file uploaded in ''Wave component file''');
                            app.WavecomponentfileEditField.Value  = "NO FILE";
                            app.WavecomponentfileEditField.FontColor = 'w';
                            app.WavecomponentfileEditField.BackgroundColor = 'r';
                            error = error + 1;
                        elseif strcmp(app.WavecomponentfileEditField.Value , app.WaveFileNAME)
                            %If the file is not in 'INPUT_FILES/Input_Data', copy it there
                            if ~exist(fullfile(app.inputFolder, app.WaveFileNAME), 'file')
                                copyfile(app.WaveFileLOC, app.inputFolder);
                            end

                        end

                    case "WK_DATA2D"
                        %---------------------
                        %......WK_DATA2D......
                        %---------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = WK_DATA2D\n WaveCompFile = Input_Data/%s\n Xc_WK = %.2f\n Yc_WK = %.2f\nDEP_WK = %.2f\nDelta_WK = %.2f\n ', app.WavecomponentfileEditField.Value,Xc,Yc,DEP,app.WidthDeltaEditField.Value);

                        %Log report
                        checkAndReport(app.XcoordinatemEditField, 'X Coordinate', fileId);
                        checkAndReport(app.YcoordinatemEditField, 'Y Coordinate', fileId);
                        checkAndReport(app.WidthDeltaEditField, 'Width Delta', fileId);
                        checkAndReport(app.WaterdepthmEditField, 'Water depth (m)', fileId);

                        %Check if the file is loaded
                        wavecompconditions = [...
                            isempty(app.WavecomponentfileEditField.Value ), ...
                            ~any(strcmp(app.WavecomponentfileEditField.Value , app.WaveFileNAME)), ...
                            str2double(app.WavecomponentfileEditField.Value ) == 0 ...
                            ];

                        if any(wavecompconditions)
                            %Display 'NO FILE' in the textbox and note the issue in the log report
                            fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:','Wavemaker:');
                            fprintf(fileId, '%-35s%-s\n', '', 'No file uploaded in ''Wave component file''');
                            app.WavecomponentfileEditField.Value  = "NO FILE";
                            app.WavecomponentfileEditField.FontColor = 'w';
                            app.WavecomponentfileEditField.BackgroundColor = 'r';
                            error = error + 1;
                        elseif strcmp(app.WavecomponentfileEditField.Value , app.WaveFileNAME)
                            % Check if the uploaded file is not in the input folder
                            if ~exist(fullfile(app.inputFolder, app.WaveFileNAME), 'file')
                                % If the file is not in 'INPUT_FILES/Input_Data', copy it there
                                copyfile(app.WaveFileLOC, app.inputFolder);
                            end
                        end

                    case "INI_GAUSSIAN"
                        %------------------------
                        %......INI_GAUSSIAN......
                        %------------------------
                        tempWavemaker2 = sprintf('WAVEMAKER = INI_GAUSSIAN\nAMP = %.2f\nXc = %.4f\nYc = %.4f\nWID = %.1f\n', AMP,Xc, Yc, WID);

                        %Log report
                        checkAndReport(app.AmplitudemEditField, 'Amplitude (m)', fileId);
                        checkAndReport(app.XcoordinatemEditField, 'X Coordinate', fileId);
                        checkAndReport(app.YcoordinatemEditField, 'Y Coordinate', fileId);
                        checkAndReport(app.WidthmEditField, 'Width (m)', fileId);

                    case "See options"
                        fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:','No Wavemaker type selected');
                        tempWavemaker2 = sprintf('!**No Wavemaker type selected\n');
                        error = error + 1;
                end
```

```

    %Combine all lines
    tempWavemaker = [tempWavemaker1, tempWavemaker2];

    %Add these lines in the input.txt
    fprintf(fileInputId, '%s', tempWavemaker);
end

%-------------------------------------------
%------INITIAL CONDITION: Meteotsunami------
%-------------------------------------------
if app.MeteotsunamiButton.Value
    tempMeteo = sprintf('\n! ----------------↙
METEOTSUNAMI----------------\n');

    %-----------------------
    %......Air Pressure......
    %-----------------------
    if strcmp(app.AirpressureSwitch.Value, "On")
        tempMeteo = [tempMeteo, sprintf('MeteoGausian = T\n')];

       %Check if the pressure file exists
        Pressureconditions = [...
            isempty(app.PressurefileEditField.Value), ... % Check↙
if the field is empty
            ~any(strcmp(app.PressurefileEditField.Value,  app.↙
PressureFileNAME)), ... % Check if the value does not match expected names
            str2double(app.PressurefileEditField.Value) == 0 ... %↙
Check if the value is zero (assuming numeric comparison is relevant)
            ];

        if any(Pressureconditions)
            %Display 'NO FILE' in the textbox and note the issue↙
in the log report
            app.PressurefileEditField.BackgroundColor = 'r';
            app.PressurefileEditField.Value = "NO FILE";
            app.PressurefileEditField.FontColor = 'w';
            tempMeteo = [tempMeteo, sprintf('METEO_GAUSIAN_FILE =↙
Input_Data/\n')];
            fprintf(fileId,'%-30s%-s\n','INITIAL↙
CONDITION:','Meteotsunami');
            fprintf(fileId,'%-35s%-s\n','','No file uploaded in↙
''Pressure file''');
            error = error + 1;
            error_mkfile = error_mkfile + 1;
            error_mkfiletext9 = sprintf('INITIAL CONDITION:↙
Meteotsunami');
        else
            tempMeteo = [tempMeteo, sprintf('METEO_GAUSIAN_FILE =↙
Input_Data/%s\n', string(app.PressureFileNAME))];
            %If the file is not in 'INPUT_FILES/Input_Data', copy↙
it there
            if ~exist(fullfile(app.inputFolder, app.↙
PressureFileNAME), 'file')
                copyfile(app.PressureFileLOC, app.inputFolder);
            end
        end
    else
```

```
                    %Set the value to False
                    tempMeteo = [tempMeteo, sprintf('MeteoGaussian = F\n')];
                end

                %--------------------------------
                %......Constant Wind Field......
                %--------------------------------
                if strcmp(app.ConstantwindfieldSwitch.Value, "On")
                    tempMeteo = [tempMeteo, sprintf('WindConstantField = T\n')];

                    %Check if the uploaded Constant wind file exists
                    WindWaveconditions = [...
                        isempty( app.ConstantwindfileEditField.Value), ... % Check if the field is empty
                        ~any(strcmp( app.ConstantwindfileEditField.Value, app.ConsWindNAME)), ... % Check if the value does not match expected names
                        str2double( app.ConstantwindfileEditField.Value) == 0 ... % Check if the value is zero (assuming numeric comparison is relevant)
                        ];

                    if any(WindWaveconditions)
                        tempMeteo = [tempMeteo, sprintf('CONSTANT_WIND_FILE = Input_Data/\n')];

                        %Display 'NO FILE' in the textbox and note the issue in the log report
                        app.ConstantwindfileEditField.BackgroundColor =  'r';
                        app.ConstantwindfileEditField.Value = "NO FILE";
                        app.ConstantwindfileEditField.FontColor = 'w';

                        fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:','Meteotsunami');
                        fprintf(fileId,'%-35s%-s\n','','No file uploaded in ''Pressure file''');
                        error = error + 1;
                        error_mkfile = error_mkfile + 1;
                        error_mkfiletext9 = sprintf('INITIAL CONDITION: Meteotsunami');
                    else
                        tempMeteo = [tempMeteo, sprintf('CONSTANT_WIND_FILE = Input_Data/%s\n', string(app.ConsWindNAME))];
                        %If the file is not in 'INPUT_FILES/Input_Data', copy it there
                        if ~exist(fullfile(app.inputFolder, app.ConsWindNAME), 'file')
                            copyfile(app.ConsWindLOC, app.inputFolder);
                        end
                    end
                else
                    %Set the value to False
                    tempMeteo = [tempMeteo, sprintf('WindConstantField = F\n')];
                end

                %----------------------------------
                %......Wind Wave Interaction......
                %----------------------------------
```

```matlab
                if strcmp(app.WindforceSwitch.Value, "On")
                    tempMeteo = [tempMeteo, sprintf('WindForce = T\nCdw = %.3g\n', app.WindstresscoefficientEditField.Value)];
                    app.WindstresscoefficientEditField.FontColor = 'k';

                    % Wind Wave Interaction settings
                    if strcmp(app.WindwaveinteractionSwitch.Value, "On")
                        tempMeteo = [tempMeteo, sprintf('WindWaveInteraction = T\nWindCrestPercent = %.2f\n', app.CrestRatioEditField.Value)];
                        app.CrestRatioEditField.FontColor = 'k';
                    else
                        %Set the value to False
                        tempMeteo = [tempMeteo, sprintf('WindWaveInteraction = F\n')];
                    end
                else
                    %Set the value to False
                    tempMeteo = [tempMeteo, sprintf('WindForce = F\n')];
                end

                %-------------------------------
                %......Holland Storm Model......
                %-------------------------------
                if strcmp(app.HollandstormmodelSwitch.Value, "On")
                    tempMeteo = [tempMeteo, sprintf('WindHollandModel = T\n')];

                    %Check if the uploaded Storm file exists
                    Stormconditions = [...
                        isempty(app.StormfileEditField.Value), ... % Check if the field is empty
                        ~any(strcmp(app.StormfileEditField.Value,app.StormFileNAME)), ... % Check if the value does not match expected names
                        str2double(app.StormfileEditField.Value) == 0 ... % Check if the value is zero (assuming numeric comparison is relevant)
                        ];

                    if any(Stormconditions)
                        tempMeteo = [tempMeteo, sprintf('STORM_FILE = Input_Data/\n')];

                        %Display 'NO FILE' in the textbox and note the issue in the log report
                        app.StormfileEditField.BackgroundColor = 'r';
                        app.StormfileEditField.Value = "NO FILE";
                        app.StormfileEditField.FontColor = 'w';

                        fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:','Meteotsunami');
                        fprintf(fileId,'%-35s%-s\n','','No file uploaded in ''Storm file''');
                        error = error + 1;
                        error_mkfile = error_mkfile + 1;
                        error_mkfiletext9 = sprintf('INITIAL CONDITION: Meteotsunami');
                    else
                        tempMeteo = [tempMeteo, sprintf('STORM_FILE = Input_Data/%s\n', string(app.StormFileNAME))];
                        % Check if the uploaded file is not from the INPUT_FOLDER
                        if ~exist(fullfile(inputFolder, app.StormFileNAME), 'file')
                            copyfile(app.StormFileLOC, inputFolder);
                        end
                    end
                else
                    %Set parameter to False
                    tempMeteo = [tempMeteo, sprintf('WindHollandModel = F\n')];
                end

                %Add these lines in the input.txt
                fprintf(fileInputId, '%s', tempMeteo);
            end



            %---------------------------------------------------
            %------INITIAL CONDITION: Sediment Transport------
            %---------------------------------------------------
            tempSediment = sprintf('\n! ---------------- SEDIMENT TRANSPORT ---------------- \n');

            if app.SedimentTransportCheckBox.Value
                %--------------------
                %......Sediment......
                %--------------------
                if app.TVDButton.Value
                    tempSediment = [tempSediment, sprintf('Sed_Scheme = TVD \n')];
                elseif app.UpwindingButton.Value
                    tempSediment = [tempSediment, sprintf('Sed_Scheme = Upwinding \n')];
                end

                editFields = [app.TimeStepSedToMorphlEditField, app.MinDepthSedPickUpEditField, ...
                    app.RungeKuttaparameter2EditField, app.RungeKuttaparameter1EditField, ...
                    app.ShieldsparamBedloadEditField, app.CriticalShieldsEditField, ...
                    app.SettlingvelocitymsEditField, app.SedimentporosityEditField, ...
                    app.SedimentdensityEditField, app.MediangrainDMEditField];

                for field1 = editFields
                    field1.FontColor = 'k';
                end

                %Median Grain diameter
                tempSediment = [tempSediment, sprintf('D50 = %.3f \n', app.MediangrainDMEditField.Value)];

                %Sediment density
                tempSediment = [tempSediment, sprintf('Sdensity = %.3f \n', app.SedimentdensityEditField.Value)];
```

```

                %Sediment porosity
                tempSediment = [tempSediment, sprintf('n_porosity = %.3f \n', app.SedimentporosityEditField.Value)];

                %Settling velocity
                tempSediment = [tempSediment, sprintf('WS = %.3f \n', app.SettlingvelocitymsEditField.Value)];

                %Shields parameters (suspended load)
                tempSediment = [tempSediment, sprintf('Shields_cr = %.3f \n', app.CriticalShieldsEditField.Value)];

                %Shields parameters (bedload)
                tempSediment = [tempSediment, sprintf('Shields_cr_bedload = %.3f \n', app.ShieldsparamBedloadEditField.Value)];

                %Runge-Kutta pameters 1 and2
                tempSediment = [tempSediment, sprintf('Kappa1 = %.4f \n', app.RungeKuttaparameter1EditField.Value)];

                tempSediment = [tempSediment, sprintf('Kappa2 = %.3f \n', app.RungeKuttaparameter2EditField.Value)];

                %Minimum depth for sediment pickup action
                tempSediment = [tempSediment, sprintf('MinDepthPickup = %.3f \n', app.MinDepthSedPickUpEditField.Value)];

                %---------------------------------
                %......Morphological Changes......
                %---------------------------------
                if app.UpdatedepthButton.Value
                    tempSediment = [tempSediment, sprintf('Bed_Change = T \nBedLoad = T\n')];

                elseif app.NobedchangeButton.Value
                    tempSediment = [tempSediment, sprintf('Bed_Change = F \n')];

                elseif app.Button_36.Value
                    tempSediment = [tempSediment, sprintf('Bed_Change = \n')];

                    %Log report
                    fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:','Sediment:');
                    fprintf(fileId,'%-35s%-s\n','','No type of Bed Change is selected');
                    app.UpdatedepthButton.FontColor = 'r';
                    app.NobedchangeButton.FontColor = 'r';
                    error = error + 1;
                end

                if app.HardbednoerosionButton.Value
                    if  app.FileEditField.Value == string(app.HardbottomNAME)
                        tempSediment = [tempSediment, sprintf('Hard_bottom = T\n')];
                        tempSediment = [tempSediment, sprintf('Hard bottom file = Input Data/%s\n', string(app.HardbottomNAME))];

                        if ~exist(fullfile(app.inputFolder, app.HardbottomNAME), 'file')
                            %If the file is not in 'INPUT_FILES/Input_Data', copy it there
                            copyfile(app.HardbottomLOC, app.inputFolder);
                        end
                    else
                        tempSediment = [tempSediment, sprintf('Hard_bottom_file = Input_Data/\n')];

                        %Display 'NO FILE' in the textbox and note the issue in the log report
                        app.FileEditField.Value = "NO FILE";
                        app.FileEditField.FontColor = 'w';
                        app.FileEditField.BackgroundColor = 'r';
                        fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:','Sediment:');
                        fprintf(fileId,'%-35s%-s\n','','No file uploaded in ''Hard Bottom File''');

                        error = error + 1;
                        error_mkfile = error_mkfile + 1;
                        error_mkfiletext11 = sprintf('Initial Condition: Sediment');

                    end
                elseif app.SedimentbedButton.Value
                    %Set value to False
                    tempSediment = [tempSediment, sprintf('Hard_bottom = F\n')];
                elseif app.Button_37.Value
                    tempSediment = [tempSediment, sprintf('Hard_bottom = \n')];

                    %Note the issue in the log report
                    fprintf(fileId,'%-30s%-s\n','INITIAL CONDITION:','Sediment:');
                    fprintf(fileId,'%-35s%-s\n','','No Bed Type is selected');
                    app.HardbednoerosionButton.FontColor = 'r';
                    app.SedimentbedButton.FontColor = 'r';
                    error = error + 1;
                end

                %----------------------
                %......Avalanche......
                %----------------------
                if app.YesButton_8.Value
                    app.RungeKuttaparam2EditField_2.FontColor = 'k';
                    tempSediment = [tempSediment, sprintf('Avalanche = T \n')];
                    tempSediment = [tempSediment, sprintf('Tan_phi = %.3g \n', app.RungeKuttaparam2EditField_2.Value)];
                end

                %Add these lines in the input.txt
                fprintf(fileInputId, '%s', tempSediment);

```

```
            end


            %-------------------------------------
            %------INITIAL CONDITION: Vessel------
            %-------------------------------------
            if app.VesselButton.Value
                %---------------------
                %......Vessel......
                %---------------------
                tempVessel1 = sprintf('\n! ---------------- SHIP WAKES ----------------------------\n');

                %Check if the file is loaded
                Vesselconditions = [...
                    isempty(app.ImportvesselfilesTextArea.Value), ...
                    ~any(strcmp( app.ImportvesselfilesTextArea.Value, app.VesselNAME)), ...
                    str2double( app.ImportvesselfilesTextArea.Value) == 0 ...
                    ];

                if any(Vesselconditions)
                    tempVessel2 = sprintf(['VESSEL_FOLDER = ./\n' ...
                        'NumVessel = \n']);
                    %Display 'NO FILE' in the textbox and note the issue in the log report
                    fprintf(fileId, '%-30s%-s\n', 'INITIAL CONDITION:', 'Vessel:');
                    fprintf(fileId, '%-35s%-s\n', '', 'No file uploaded in ''Import vessel file/s:''');
                    app.ImportvesselfilesTextArea.Value = "NO FILE";
                    app.ImportvesselfilesTextArea.FontColor = 'w';
                    app.ImportvesselfilesTextArea.BackgroundColor = 'r';

                    error = error + 1;
                    error_mkfile = error_mkfile + 1;
                    error_mkfiletext8 = sprintf('Initial Condition: Vessel');
                else
                    tempVessel2 = sprintf(['VESSEL_FOLDER = %s\n' ...
                        'NumVessel = %s\n'], fullfile('.', 'Input_Data/'), num2str(app.VesselNumber));

                    for g = 1:length(app.ImportvesselfilesTextArea.Value)
                         %Copy files to'INPUT_FILES/Input_Data'
                        copyfile(app.VesselLOC(g), app.inputFolder);
                    end
                end

                %Combine all the lines
                tempVessel = [tempVessel1,tempVessel2];

                if app.ActivateCheckBox.Value
                    %------------------------------
                    %......Deep Draft Vessels......
                    %------------------------------
                    %Options for Methods:

                    %Friction
                    if app.FrictionCheckBox_2.Value
                        frictionCoefficient = app.FrictionCoefficientEditField.Value;
                        tempVessel = [tempVessel, sprintf('FrictionMethod = T\nCdDeepDraft = %.1f\n', frictionCoefficient)];
                        app.FrictionCoefficientEditField.FontColor =  'k';
                    else
                        tempVessel = [tempVessel, sprintf('FrictionMethod = F\n')];
                    end

                    %Viscosity
                    if app.ViscosityCheckBox.Value
                        viscosityValue = app.ValueEditField.Value;
                        tempVessel = [tempVessel, sprintf('ViscosityMethod = T\nVisDeepDraft = %.1f\n', viscosityValue)];
                        app.ValueEditField.FontColor =  'k';
                    else
                        tempVessel = [tempVessel, sprintf('ViscosityMethod = F\n')];
                    end

                    %Shock capturing
                    if app.ShockcapturingCheckBox.Value
                        tempVessel = [tempVessel, sprintf('MaskMethod = T\n')];
                    else
                        tempVessel = [tempVessel, sprintf('MaskMethod = F\n')];
                    end

                    minimumClearance = app.MinimumClearanceEditField.Value;

                    %Add another line at the end of tempVessel
                    tempVessel = [tempVessel, sprintf('CLEARANCE = %.1f\n', minimumClearance)];
                    app.MinimumClearanceEditField.FontColor =  'k';
                end

                %Add these lines in the input.txt
                fprintf(fileInputId, '%s', tempVessel);
            end

            %---------------------------------------
            %------PERIODIC BOUNDARY CONDITION------
            %---------------------------------------
            tempPBC = sprintf([
                '\n ! ---------------- PERIODIC BOUNDARY CONDITION --------- \n' ...
                '   ! South-North periodic boundary condition\n' ...
                'PERIODIC = F\n']);

            %Add these lines in the input.txt
            fprintf(fileInputId, '%s',tempPBC);

            %---------------------------
            %------SPONGE BOUNDARY------
            %---------------------------
```

```
            tempSponge = sprintf([
                '\n  ! ---------------- SPONGE LAYER ------------------------\n' ...
                '  ! DHI type sponge layer\n' ...
                '  ! need to specify widths of four boundaries and parameters\n' ...
                '  ! set width=0.0 if no sponge\n' ...
                '  ! R_sponge: decay rate\n' ...
                '  ! A_sponge: maximum decay rate\n' ...
                '  ! e.g., sharp: R=0.85 \n' ...
                '  !       mild:  R=0.90, A=5.0 \n' ...
                '  !       very mild, R=0.95, A=5.0 \n']);


            %Add these lines in the input.txt
            fprintf(fileInputId, '%s',tempSponge);

            % Set sponge dimensions if any sponge setting is active
            if app.DirectCheckBox.Value || app.FrictionCheckBox.Value || app.DiffusionCheckBox.Value
                spongeOn = sprintf('SPONGE_ON = T \n');
            else
                spongeOn = sprintf('SPONGE_ON = F \n');
            end
            fprintf(fileInputId, '%s',spongeOn);

            %Direct Sponge
            if app.DirectCheckBox.Value
                fprintf(fileInputId, 'DIRECT_SPONGE = T \n');

                % Determine the decay rate and associated sponge settings
                switch app.DecayTypeDropDown.Value
                    case 'Very mild'
                        decayRate = sprintf([ ...
                            'R_sponge = 0.95 \n' ...
                            'A_sponge = 5.0 \n']);
                    case 'Mild'
                        decayRate = sprintf([ ...
                            'R_sponge = 0.9 \n' ...
                            'A_sponge = 5.0 \n']);
                    case 'Sharp'
                        decayRate = sprintf('R_sponge = 0.85 \n');
                end

                %Add these lines in the input.txt
                fprintf(fileInputId, '%s', decayRate);
            end

            %Friction
            spongeSetting = '';
            if app.FrictionCheckBox.Value
                spongeSetting = sprintf('FRICTION_SPONGE = T\nCDsponge = %.3f\n', app.MaximumCdEditField.Value);
                %Add these lines in the input.txt
                fprintf(fileInputId, '%s', spongeSetting);
            end

            %Diffusion
```

```matlab
        if app.DiffusionCheckBox.Value
            spongeSetting = sprintf('DIFFUSION_SPONGE = T\nCsp = %.3f \n', app.MaximumCspEditField.Value);
            %Add these lines in the input.txt
            fprintf(fileInputId, '%s', spongeSetting);
        end

        %Add these lines in the input.txt
        % fprintf(fileInputId, '%s', spongeSetting);


        % Set sponge dimensions if any sponge setting is active
        if app.DirectCheckBox.Value || app.FrictionCheckBox.Value || app.DiffusionCheckBox.Value
            spongeBound = sprintf([
                'Sponge_west_width = %.1f\n' ...
                'Sponge_east_width = %.1f\n' ...
                'Sponge_south_width = %.1f\n' ...
                'Sponge_north_width = %.1f\n'], ...
                app.WestEditField.Value, app.EastEditField.Value, app.SouthEditField.Value, app.NorthEditField.Value);

            %Log warnings when the sponge layer is turned ON but the boundaries are set to zero
            if app.NorthEditField.Value == 0
                app.NorthEditField.FontColor = 'r';
                fprintf(fileId,'%-30s%-s\n','SPONGE BOUNDARY:', 'North buffer is zero');
            end
            if app.EastEditField.Value == 0
                app.EastEditField.FontColor = 'r';
                fprintf(fileId,'%-30s%-s\n','SPONGE BOUNDARY:', 'East buffer is zero');
            end
            if app.WestEditField.Value == 0
                app.WestEditField.FontColor = 'r';
                fprintf(fileId,'%-30s%-s\n','SPONGE BOUNDARY:', 'West buffer is zero');
            end
            if app.SouthEditField.Value == 0
                app.SouthEditField.FontColor = 'r';
                fprintf(fileId,'%-30s%-s\n','SPONGE BOUNDARY:', 'South buffer is zero');
            end

        else
            spongeBound = sprintf([
                'Sponge_west_width = 0.0\n' ...
                'Sponge_east_width = 0.0\n' ...
                'Sponge_south_width = 0.0\n' ...
                'Sponge_north_width = 0.0\n']);

            app.NorthEditField.FontColor = '0.65,0.65,0.65';
            app.EastEditField.FontColor = '0.65,0.65,0.65';
            app.WestEditField.FontColor = '0.65,0.65,0.65';
            app.SouthEditField.FontColor = '0.65,0.65,0.65';
        end

        %Add these lines in the input.txt
        fprintf(fileInputId, '%s', spongeBound);

        %---------------------
        %------OBSTACLES------
        %---------------------
        if app.AddobstaclesCheckBox.Value
            if isempty(app.ObstacleFilename.Value) || ~strcmp(app.ObstacleFilename.Value, app.obstacleNAME) || ~exist(app.obstacleLOC, "file")
                % Display 'NO FILE' in the textbox
                app.ObstacleFilename.Value = "NO FILE";
                app.ObstacleFilename.FontColor = 'w';
                app.ObstacleFilename.BackgroundColor = 'r';
                fprintf(fileId, '%-30s%-s\n', 'OBSTACLES:', 'No file is uploaded');
                error = error + 1;
            else
                if strcmp(app.ObstacleFilename.Value, string(app.obstacleNAME))
                    tempObstacle = sprintf([
                        '  \n! ----------------OBSTACLES-----------------------------\n' ...
                        '  ! obstacle structures using mask_struc file\n' ...
                        '  ! mask_struc =0 means structure element\n' ...
                        '  ! give a file contains a mask array with Mloc X Nloc\n' ...
                        '  OBSTACLE_FILE= %s\n'], ...
                        app.ObstacleFilename.Value);

                    % Add these lines in the input.txt
                    fprintf(fileInputId, '%s', tempObstacle);

                    if exist(app.obstacleLOC, "file")
                        obstacle = readmatrix(app.obstacleLOC);
                        if isnumeric(obstacle)
                            if ~all(obstacle(:) == 0 | obstacle(:) == 1)
                                fprintf(fileId, '%-30s%-s\n', 'OBSTACLES:', 'File should only contain values of 1 and 0');
                            end
                        end
                    else
                        fprintf(fileId, '%-30s%-s\n', 'OBSTACLES:', 'Uploaded file is empty');
                    end

                    % If the file is not in 'INPUT_FILES/Input_Data', copy it there
                    if ~exist(fullfile(inputFolder, app.obstacleNAME), 'file')
                        copyfile(app.obstacleLOC, inputFolder);
                    end
                end
            end
        end

        %-------------------
        %------PHYSICS------
```

```
            %-------------------
            tempPhysics1 = sprintf([
                '\n  ! ----------------PHYSICS------------------------------\n' ...
                '  ! parameters to control type of equations\n' ...
                '  ! dispersion: all dispersive terms\n' ...
                '  ! gamma1=1.0,gamma2=0.0: NGs equations\n' ...
                '  ! gamma1=1.0,gamma2=1.0: Fully nonlinear equations\n']);

            %Initial set up for the second part of the lines for this section
            tempPhysics2 = '';

            %-----------------------------------------
            %......Linear Shallow Water Equation......
            %-----------------------------------------
            if app.LinearShallowWaterEquationButton.Value
                app.LinearShallowWaterEquationButton.FontColor = 'k';
                tempPhysics2 = sprintf([
                    'Gamma1 = 0.0\n' ...
                    'Gamma2 = 0.0\n' ...
                    'Gamma3 = 0.0\n']);

                %--------------------------------------------
                %......Nonlinear Shallow Water Equation......
                %--------------------------------------------
            elseif app.NonlinearShallowWaterEquationButton.Value
                app.NonlinearShallowWaterEquationButton.FontColor = 'k';
                tempPhysics2 = sprintf([
                    'Gamma1 = 0.0\n' ...
                    'Gamma2 = 0.0\n' ...
                    'Gamma3 = 1.0\n' ...
                    'Beta_ref=-0.531\n']);

                %--------------------------------------
                %......Nonlinear Boussinesq Equation......
                %-----------------------------------------
            elseif app.FullyNonlinearBoussinesqEquationButton.Value
                app.FullyNonlinearBoussinesqEquationButton.FontColor = 'k';
                tempPhysics2 = sprintf([
                    'Gamma1 = 1.0\n' ...
                    'Gamma2 = 1.0\n' ...
                    'Gamma3 = 1.0\n' ...
                    'Beta_ref=-0.531\n' ...
                    'SWE_ETA_DEP = 0.8\n'...
                    'DISPERSION = T\n']);
            end



            if app.LinearShallowWaterEquationButton.Value || app.FullyNonlinearBoussinesqEquationButton.Value || app.NonlinearShallowWaterEquationButton.Value
                %Combine all the lines
                tempPhysics = [tempPhysics1 tempPhysics2];

                %Add these lines in the input.txt
                fprintf(fileInputId, '%s', tempPhysics);
            elseif app.Button.Value
                %Display 'NO FILE' in the textbox and note the issue in the log report
                app.FullyNonlinearBoussinesqEquationButton.FontColor = 'r';
                app.LinearShallowWaterEquationButton.FontColor = 'r';
                app.NonlinearShallowWaterEquationButton.FontColor = 'r';
                fprintf(fileId,'%-30s%-s\n','PHYSICS:', 'No equation is selected');
                error = error + 1;
            end

            %---------------------------
            %------BOTTOM FRICTION------
            %----------------------------
            % Initialize base friction information
            tempFriction1 = sprintf(' \n !---------------Friction---------------------------\n');

            tempFriction2 = '';  %Initial set up for the second part of the lines for this section

            if app.ConstantCoefficientButton.Value
                %-----------------------------
                %......Constant Friction......
                %-----------------------------
                tempFriction2 = sprintf([
                    'Friction_Matrix= F\n' ...
                    'Cd = %.3f\n' ...
                    'Cd_fixed = %.3f\n'], ...
                    app.ConstantFrictionValue.Value, ...
                    app.ConstantFrictionValue.Value);

                if app.ConstantFrictionValue.Value == 0
                    tempFriction2 = sprintf('Friction_Matrix= F \n  Cd = 0.0');
                end

            elseif app.NonconstantButton.Value
                %-------------------------------
                %......Nonconstan Friction......
                %-------------------------------
                if isempty(app.FrictionFileEditField.Value) || ~strcmp(app.FrictionFileEditField.Value, app.filenamefriction) || ~exist(app.Cd_file,"file")
                    %Display 'NO FILE' in the textbox and note the issue in the log report
                    app.FrictionFileEditField.BackgroundColor = 'r';
                    app.FrictionFileEditField.Value = "NO FILE";
                    app.FrictionFileEditField.FontColor = 'w';
                    fprintf(fileId, '%-30s%-s\n', 'BOTTOM FRICTION:', 'No file is uploaded');

                    error = error + 1;
                    error_mkfile = error_mkfile + 1;
                    error_mkfiletext7 = sprintf('Friction');
                else
                    tempFriction2 = sprintf('Friction_Matrix= T\nCd_file= Input_Data/%s\n', app.FrictionFileEditField.Value);

                    if ~exist(fullfile(app.inputFolder, app.Cd_file), 'file')
```

```
                    % If the file is not in 'INPUT_FILES/Input_Data', copy it there
                    copyfile(app.Cd_file, app.inputFolder);
                end
            end
        end

        % Combine the parts
        tempFriction = [tempFriction1 tempFriction2];

        %Add these lines in the input.txt
        fprintf(fileInputId, '%s', tempFriction);

        %--------------------
        %------NUMERICS------
        %--------------------
        if app.LinearShallowWaterEquationButton.Value || app.NonlinearShallowWaterEquationButton.Value
            tempNumerics =  sprintf([
                '\n  ! ----------------NUMERICS----------------------------\n'...
                'HIGH_ORDER = SECOND \n' ...
                'CFL = 0.5 \n' ...
                '  ! Froude Number Cap (to avoid jumping drop, set 10) \n' ...
                'FroudeCap = %.2f \n'],app.FroudeCapEditField.Value);
        else
            tempNumerics =  sprintf([
                '\n  ! ----------------NUMERICS----------------------------\n'...
                'HIGH_ORDER = THIRD \n' ...
                'CFL = 0.5 \n' ...
                '  ! Froude Number Cap (to avoid jumping drop, set 10) \n' ...
                'FroudeCap = %.2f \n'],app.FroudeCapEditField.Value);
        end

        %Add these lines in the input.txt
        fprintf(fileInputId, '%s', tempNumerics);

        %-------------------------
        %------MINIMUM DEPTH------
        %-------------------------
        tempWetDry1 = sprintf('\n  ! --------------WET-DRY-------------------------------\n');

        tempWetDry2 = sprintf([
            '  ! MinDepth for wetting-drying\n' ...
            'MinDepth= %.2f\n'], ...
            app.WetDrySchemeEditField.Value);

        if app.ConstantCoefficientButton.Value || app.NonconstantButton.Value
            tempWetDry3 = sprintf([
                '  ! MinDepthfrc to limit bottom friction\n' ...
                'MinDepthFrc = %.2f \n'], ...
                app.BottomFrictionEditField.Value);

            tempWetDry = [tempWetDry1, tempWetDry2, tempWetDry3];
        else
            tempWetDry = [tempWetDry1, tempWetDry2];
        end

        %Add these lines in the input.txt
        fprintf(fileInputId, '%s', tempWetDry);

        %-------------------------
        %-------WAVE BREAKING------
        %-------------------------
        if app.FullyNonlinearBoussinesqEquationButton.Value
            tempShowBreaking1 = sprintf([
                '\n  ! -------------- SHOW BREAKING ------------------------\n' ...
                '  ! breaking is calculated using shock wave capturing scheme\n' ...
                '  ! the criteria is only for demonstration or bubble calculation\n' ...
                '  ! Cbrk1=0.45,Cbrk2=0.35, for irregular waves, there are much small!\n']);

            if app.ShockwavecapturingButton.Value
                tempShowBreaking2 = sprintf([
                    'SHOW_BREAKING = T\n' ...
                    'VISCOSITY_BREAKING = F\n']);
            elseif app.ViscositybreakingButton.Value
                tempShowBreaking2 = sprintf([
                    'SHOW_BREAKING = F\n' ...
                    'VISCOSITY_BREAKING = T\n'...
                    'Cbrk1 = %.2f\n' ...
                    'Cbrk2 = %.2f\n'], ...
                    app.C1EditField.Value, ...
                    app.C2EditField.Value);
            else
                tempShowBreaking2 = '';
            end

            tempShowBreaking = [tempShowBreaking1, tempShowBreaking2];
            fprintf(fileInputId, '%s', tempShowBreaking);
        end

        if app.AddrollereffectsCheckBox.Value && ~app.FullyNonlinearBoussinesqEquationButton.Value
            tempShowBreaking1 = sprintf([
                '\n  ! -------------- SHOW BREAKING ------------------------\n']);
            tempShowBreaking2 = sprintf('ROLLER_EFFECT = T');
            app.UndertowCheckBox.Value = 1; %Set the expected output UNDERTOW = T
            app.RollerinducedFluxCheckBox.Value = 1; %Set the expected output Roller = T

            %Combine all lines
            tempShowBreaking = [tempShowBreaking1, tempShowBreaking2];

            %Add these lines in the input.txt
            fprintf(fileInputId, '%s\n', tempShowBreaking);
```

```matlab

            elseif app.AddrollereffectsCheckBox.Value && app.FullyNonlinearBoussinesqEquationButton.Value
                fprintf(fileInputId, '%s\n','ROLLER_EFFECT = T');
            end

            %------------------
            %------MIXING------
            %------------------
            if app.UseSmagorinskyCheckBox.Value
                tempMixing = sprintf([...
                    '\n  ! ----------------- MIXING ---------------------------\n' ...
                    '  ! if use smagorinsky mixing, have to set -DMIXING in Makefile\n' ...
                    '  ! and set averaging time interval, T_INTV_mean, default: 20s\n' ...
                    'T_INTV_mean = %.2f\n' ...
                    'STEADY_TIME = %.2f\n' ...
                    'C_smg = 0.25\n'], ...
                    app.TimeintervalsecEditField.Value, ...
                    app.SteadyTimeEditField.Value);
                fprintf(fileInputId, '%s', tempMixing);
            end

            %---------------------------
            %------EXPECTED OUTPUT------
            %---------------------------
            tempOutput1 = sprintf('\n! ----------------- OUTPUT-----------------------------\n');

            %------------------
            %------GAUGES------
            %------------------
            if app.ImportgaugelistCheckBox.Value
                %Use the loaded file as regular virtual gauges, not to be used for nested grid

                %Check if gauge file is loaded
                loadFileNum = str2double(app.LoadFileEditField.Value);

                conditions = [...
                    isempty(app.LoadFileEditField.Value), ...
                    loadFileNum == 0, ...
                    ~any(strcmp(app.LoadFileEditField.Value,app.GAUGEFILE)) ...
                    ];

                if any(conditions)
                    tempOutput2 = sprintf('NumberStations = 0\n');

                    %Display 'NO FILE' in the textbox and note the issue in the log report
                    app.LoadFileEditField.BackgroundColor = 'r';
                    app.LoadFileEditField.Value = "NO FILE";
                    app.LoadFileEditField.FontColor = 'w';
                    fprintf(fileId,'%-30s%-s\n','GAUGES:', 'No file is uploaded');
                    tempOutput = sprintf('NumberStations = 0\n');
                    error = error + 1;

                    app.SpacingEditField.FontColor = '0.65,0.65,0.65';
                    app.WestEditField_2.FontColor = '0.65,0.65,0.65';
                    app.SouthEditField_2.FontColor = '0.65,0.65,0.65';
                    app.EastEditField_2.FontColor = '0.65,0.65,0.65';
                    app.NorthEditField_2.FontColor = '0.65,0.65,0.65';

                elseif app.LoadFileEditField.Value == string(app.GAUGEFILE)
                    tempOutput2 = sprintf([
                        'NumberStations = %s\n' ...
                        'STATIONS_FILE =  Input_Data/%s\n'], ...
                        app.NumberStations, ...
                        app.STATION_FILE);

                    if contains(app.LoadFileEditField.Value, '.shp')
                        % If the file type is a shapefile, convert it to a tab-delimited text file
                        outputTEXTFILE = fullfile(app.inputFolder, string(app.GaugeFname) + '.txt');
                        writematrix(app.GaugeListGrids, outputTEXTFILE, 'Delimiter', '\t');

                    elseif contains(app.LoadFileEditField.Value, '.txt')
                        % Check if all items are numeric
                        if isnumeric(app.gaugetext0)
                            % Log Report
                            if size(app.gaugetext0, 2) ~= 2
                                fprintf(fileId, '%-30s%-s\n', 'GAUGES:', 'Make sure that the .txt file contains 2 columns (Lat Long)');
                                error = error + 1;
                            end

                            if any(isnan(app.gaugetext0(:)))
                                fprintf(fileId, '%-30s%-s\n', 'GAUGES:', 'Text file contains NaN values');
                                error = error + 1;
                            end
                        else
                            if ischar(app.gaugetext0) || isstring(app.gaugetext0)
                                if any(isletter(app.gaugetext0))
                                    fprintf(fileId, '%-30s%-s\n', 'GAUGES:', 'Uploaded .txt file contains non-numeric values');
                                    error = error + 1;
                                end
                            end
                        end

                        % If the file is not in 'INPUT_FILES/Input_Data', copy it there
                        if ~exist(fullfile(inputFolder, app.GAUGEFILE), 'file')
                            copyfile(app.GAUGEFILElocation, inputFolder);
                        end
                    end
                end
```

```
            else
                %Set value to zero
                tempOutput2 = sprintf('NumberStations = 0\n');
            end



            if app.CreatenestedgridboundaryCheckBox.Value
                %Use the gauge stations as the boundary of the nested grid

                %Create nested grid boundary code from Shi et al. (2012)
                % FUNWAVE version 3.6: simple cases > nesting tools > make_stations_single_block.m

                dx= app.SpacingEditField.Value;
                id_count=0;
                icount=0;

                x1 =  app.WestEditField_2.Value + (dx/2);
                x2 =  app.EastEditField_2.Value;
                y1 =  app.SouthEditField_2.Value + (dx/2);
                y2 =  app.NorthEditField_2.Value;


                % if app.CoupMatrixON == 0 %Use the coordinates of the uploaded .tif file
                %      %Assuming the coordinate of the SW corner of the tif file is measured at the pixel's corner.
                %      %This reads the coordinate at the centre of the pixel
                %     x1 = app.WestEditField_2.Value; %+ (dx/2);
                %     x2 = app.EastEditField_2.Value; %- (dx/2);
                %     y1 =  app.SouthEditField_2.Value; %+ (dx/2);
                %     y2 =  app.NorthEditField_2.Value; %- (dx/2);
                % end

                %Create the list of coordinates
                x=[x1:dx:x2];
                y=[y1:dx:y2];


                XCouplingLength = length(x);
                YCouplingLength = length(y);
                id_count=id_count+1;

                %Arrangement of E,W,S,N is based on the requirement of convert.f
                for j=1:length(y)
                    icount=icount+1;
                    app.NestStation(icount,1)=y(j);
                    app.NestStation(icount,2)=x(end);
                    app.NestStation(icount,3)=id_count;
                end

                id_count=id_count+1;
                % west
                for j=1:length(y)
                    icount=icount+1;
                    app.NestStation(icount,1)=y(j);
                    app.NestStation(icount,2)=x(1);
                    app.NestStation(icount,3)=id_count;
                end


                id_count=id_count+1;
                % south
                for i=1:length(x)
                    icount=icount+1;
                    app.NestStation(icount,1)=y(1);
                    app.NestStation(icount,2)=x(i);
                    app.NestStation(icount,3)=id_count;
                end

                id_count=id_count+1;
                % north
                for i=1:length(x)
                    icount=icount+1;
                    app.NestStation(icount,1)=y(end);
                    app.NestStation(icount,2)=x(i);
                    app.NestStation(icount,3)=id_count;
                end

                fullFilePathNest = fullfile(app.inputFolder, 'nestingboundary.txt');
                icount_last=icount;


                fid2 = fopen(fullFilePathNest, 'wt');
                fprintf(fid2, ['%f %f %d', '\n'], app.NestStation');
                fclose(fid2);


                tempOutput2 = sprintf([
                    'NumberStations = %d\n' ...
                    'STATIONS_FILE = Input_Data/nestingboundary.txt \n'], ...
                    icount_last);

                roundup = ceil(icount_last / 500) * 500; %round up to the nearest 500;


                %Setting up the info to be displayed in the log report
                formattedCouplingDetails = sprintf([
                    '%-30s%-s\n', ...
                    '%-30s%-s\n', ...
                    '%-30s%-s\n', ...
                    '%-30s%-s\n', ...
                    '%-30s%-s\n', ...
                    '%-30s%-s\n', ...
                    '%-30s%-s\n\n'], ...
                    'GAUGES:', '''nestingboundary.txt'' is created', ...
                    'COUPLING:', 'Details to input in convert.f for your next nested grid run:', ...
                    '', ['nsta_tot and Nb:  ' num2str(roundup)], ...
                    '', ['nsta:  ' num2str(icount_last)], ...
                    ' ', ['neast and nwest: ' num2str(YCouplingLength)], ... %num2str(app.CouplingX)], ...  %refers to the N matrix dimension
                    ' ', ['nnorth and nsouth: ' num2str(XCouplingLength)], ...%, ... %num2str(app.CouplingY)])  %refers to the M matrix dimension
                    '','Ntotal refers to the total number of rows contained in each output ‘sta_’ file generated after running your model',...
                    '','Update ''fdir'' to the directory path where your ‘sta_’ files and convert.f file are located',...
                    '','The ''convert.f'' file is located in benchmarks/sph_nesting/make_nest_file');

                %Formatting
                fprintf(fileId, formattedCouplingDetails);

                %Log Report
                fields = {
                    app.SpacingEditField, 'Gauge spacing is zero';
                    app.WestEditField_2, 'West grid coordinate is zero';
                    app.EastEditField_2, 'East grid coordinate is zero';
                    app.SouthEditField_2, 'South grid coordinate is zero';
                    app.NorthEditField_2, 'North grid coordinate is zero'
                    };

                %Loop through each field and check its value
                for i = 1:size(fields, 1)
                    field = fields{i, 1};
                    message = fields{i, 2};

                    if field.Value == 0
                        field.FontColor = 'r';
                        fprintf(fileId, '%-30s%-s\n', 'GAUGES:', message);
                        error = error + 1;
                    end
                end


            end

            %Combine all the lines and add them in the input.txt
            tempOutput = [tempOutput1,tempOutput2];
            fprintf(fileInputId, '%s', tempOutput);


            %----------------------------
            %------Expected OUTPUT------
            %----------------------------
            commands = struct( ...
                'DepthCheckBox', 'DEPTH_OUT = T \n', ...
                'UVelocityCheckBox', 'U = T \n', ...
                'VVelocityCheckBox', 'V = T \n', ...
                'UVelocitymeanCheckBox', 'Umean = T \n', ...
                'VVelocitymeanCheckBox', 'Vmean = T \n', ...
                'SurfaceElevationattimetCheckBox', 'ETA = T \n', ...
                'WetdrymaskCheckBox', 'MASK = T \n', ...
                'WetdrymaskforBoussinesqNSWECheckBox', 'MASK9 = T \n', ...
                'XSourceCheckBox', 'SourceX = T \n', ...
                'YSourceCheckBox', 'SourceY = T \n', ...
                'PMomentumFluxCheckBox', 'P = T \n', ...
                'QmomentumFluxCheckBox', 'Q = T \n', ...
                'FxFluxCheckBox', 'Fx = T \n', ...
```

```
    'GxFluxCheckBox', 'Gx = T \n', ...
    'GyFluxCheckBox', 'Gy = T \n', ...
    'BreakingAgeCheckBox', 'AGE = T \n', ...
    'MaximumWaveHeightCheckBox', 'Hmax = T \n', ...
    'FyFluxCheckBox', 'Fy = T \n', ...
    'MinimumWaveHeightCheckBox', 'Hmin = T \n', ...
    'MaximumVelocityCheckBox', 'Umax = T \n', ...
    'UndertowCheckBox', 'UNDERTOW = T \n', ...
    'MaxVorticityCheckBox', 'VORmax = T \n', ...
    'MaxMomentumFluxCheckBox', 'MFmax = T \n', ...
    'ArrivalCheckBox', 'OUT_Time = T \n', ...
    'WaveHeightCheckBox', 'WaveHeight = T \n', ...
    'RollerinducedFluxCheckBox', 'ROLLER = T \n', ...
    'PressureFieldCheckBox', 'OUT_METEO = T \n', ...
    'BreakingLocationCheckBox', 'OUT_NU = T \n' ...
    );


%Loop through each item. If the values are checked in the app, list each variable and set them to 'T' in the input.txt
fields = fieldnames(commands);
for i = 1:length(fields)
    fieldName = fields{i};
    if app.(fieldName).Value
        fprintf(fileInputId, commands.(fieldName));
    end
end

%--------------------------
%------GENERAL ERRORS------
%--------------------------
if error ~=0 %Errors are present
    %Updating the log report
    fprintf(fileId, '%s\n', '');
    fprintf(fileId, '%s\n','> Status: Creating input.txt... Completed WITH ISSUES');
    fprintf(fileId, '%s\n','> Resolve the warnings above to prevent modeling errors later');
    app.Preview_LogReport.Value = fileread(LogFile); %show the log report
    drawnow

else
    %Updating the log report
    fprintf(fileId, '%s\n', '');
    fprintf(fileId, '%s\n\n','> Status: Creating input.txt... Succesfully completed!');
    app.Preview_LogReport.Value = fileread(LogFile); %show the log report
    drawnow
end

%---------------------------
%------MAKEFILE ERRORS------
%---------------------------
%Checks if there are issues associated with Makefile creation
if error_mkfile ~= 0
    %Updating the log report
```

```matlab
                fprintf(fileId, '%s\n','> Status: The Makefile section will be enabled once the raised issues are resolved');
                fprintf(fileId, '%s\n','  Specifically, check the following sections:');

                % Define a list of variables to check
                variablesToCheck = {'error_mkfiletext3', 'error_mkfiletext5', 'error_mkfiletext7', 'error_mkfiletext8', 'error_mkfiletext9', 'error_mkfiletext11'};
                flags = {'', '', '', '', 'flag10', ''};  % Associated flags for each variable

                % Loop through each variable and check if it exists
                for i = 1:length(variablesToCheck)
                    variableName = variablesToCheck{i};
                    flag = flags{i};

                    % Use eval to check if variable exists and is not empty
                    if eval(sprintf('exist(''%s'', ''var'') && ~isempty(%s)', variableName, variableName))
                        % Print the variable's value
                        fprintf(fileId, '\t%2s\n', eval(variableName));

                        % Check and print associated flag if it exists and is not empty
                        if ~isempty(flag) && eval(sprintf('exist(''%s'', ''var'') && ~isempty(%s)', flag, flag))
                            fprintf(fileId, '\t%2s\n', eval(flag));
                        end
                    end
                end

                %Deactivate the Makefile section
                app.Mpif90EditField.Enable = 'off';
                app.Mpif90EditFieldLabel.Enable = 'off';
                app.Button_41.Enable = 'off';
                app.CreateMakefileButton.Enable = 'off';
                app.DoublePrecisionCheckBox.Enable = 'off';
                app.IntelCompilerCheckBox.Enable = 'off';
                app.ParallelModeCheckBox.Enable = 'off';
                app.MakefileEditField.Enable = 'off';
                app.MakefileEditFieldLabel.Enable = 'off';
                app.Button_42.Enable = 'off';
            else
                %Updating the log report
                fprintf(fileId, '%s\n', '');
                fprintf(fileId, '%s\n\n','> Status: The Makefile section is now enabled');

                app.CreateMakefileButton.Enable = 'on';
                app.DoublePrecisionCheckBox.Enable = 'on';
                app.IntelCompilerCheckBox.Enable = 'on';
                app.ParallelModeCheckBox.Enable = 'on';
                app.MakefileEditField.Enable = 'on';
                app.MakefileEditFieldLabel.Enable = 'on';
                app.Button_42.Enable = 'on';
            end

            %Close the log report and show it in the preview panel
            fclose(fileId);
            app.Preview_LogReport.Value = fileread(LogFile);

            %Close the input.txt and show it in the preview panel
            fclose(fileInputId);
            fileContent = fileread(fileInputPath);
            app.Preview_Input.Value = fileContent;

            drawnow
        end

        % Callback function
        function ImportButton_4Pushed(app, event)

            %
        end

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

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

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

        end

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

        end

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

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


        end

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

        % Callback function
```

```matlab
        function ButtonGroupSelectionChanged(app, event)
            selectedButton = app.ButtonGroup.SelectedObject;
            if selectedButton == app.SphericaldegreesButton
                app.SphericaldegreesButton.FontColor = 'k';
                app.CartesianmetersButton.FontColor = '0.50,0.50,0.50';
                value1 = app.XEditField;
                value2 = app.YEditField;
                value3 = '1.0';
                value4 = '1.0';
            elseif selectedButton == app.CartesianmetersButton
                app.CartesianmetersButton.FontColor = 'k';
                app.SphericaldegreesButton.FontColor = '0.50,0.50,0.50';
                value3 =  app.XEditField;
                value4=  app.YEditField;
                value1 = '1.0';
                value2 = '1.0';
            end

            app.Dphi = value1;
            app.Dtheta = value2;
            app.DX = value3;
            app.DY = value4;

        end

        % Callback function
        function ButtonGroup_2SelectionChanged(app, event)

            selectedButton = app.ButtonGroup_2.SelectedObject;
            app.SWE = selectedButton;

        end

        % Selection changed function: ButtonGroup_3
        function ButtonGroup_3SelectionChanged(app, event)
            app.FullyNonlinearBoussinesqEquationButton.FontColor = 'k';
            app.LinearShallowWaterEquationButton.FontColor = 'k';
            app.NonlinearShallowWaterEquationButton.FontColor = 'k';


            if app.FullyNonlinearBoussinesqEquationButton.Value
                app.ShockwavecapturingButton.Enable = "on";
                app.ViscositybreakingButton.Enable = "on";
                drawnow
                if   app.ViscositybreakingButton.Value
                    app.C2EditField.Enable = "on";
                    app.C2EditFieldLabel.Enable = "on";
                    app.C1EditField.Enable = "on";
                    app.C1EditFieldLabel.Enable = "on";
                end
            else
                app.ShockwavecapturingButton.Enable = "off";
                app.ViscositybreakingButton.Enable = "off";
                app.C2EditField.Enable = "off";
                app.C2EditFieldLabel.Enable = "off";
                app.C1EditField.Enable = "off";
                app.C1EditFieldLabel.Enable = "off";
            end
        end

        % Selection changed function: ButtonGroup_4
        function ButtonGroup_4SelectionChanged(app, event)
            app.SphericalButton.FontColor = 'k';
            app.CartesianButton.FontColor = 'k';

            if app.SphericalButton.Value
                app.LatitudeEditField.Enable = 'on';
                app.LatitudeEditFieldLabel.Enable = 'on';
                app.LongitudeEditField.Enable = 'on';
                app.LongitudeEditFieldLabel.Enable = 'on';
            else
                app.LatitudeEditField.Enable = 'off';
                app.LatitudeEditFieldLabel.Enable = 'off';
                app.LongitudeEditField.Enable = 'off';
                app.LongitudeEditFieldLabel.Enable = 'off';
                app.LatitudeEditField.FontColor = '0.65,0.65,0.65';
                app.LongitudeEditField.FontColor  = '0.65,0.65,0.65';
            end

        end

        % Value changed function: DepthCheckBox
        function DepthCheckBoxValueChanged(app, event)

        end

        % Value changed function: UVelocityCheckBox
        function UVelocityCheckBoxValueChanged(app, event)


        end

        % Value changed function: VVelocityCheckBox
        function VVelocityCheckBoxValueChanged(app, event)

        end

        % Value changed function: SurfaceElevationattimetCheckBox
        function SurfaceElevationattimetCheckBoxValueChanged(app, event)

        end

        % Value changed function: WetdrymaskCheckBox
        function WetdrymaskCheckBoxValueChanged(app, event)

        end

        % Value changed function: WetdrymaskforBoussinesqNSWECheckBox
        function WetdrymaskforBoussinesqNSWECheckBoxValueChanged(app, event)

        end

        % Value changed function: XSourceCheckBox
        function XSourceCheckBoxValueChanged(app, event)

        end

        % Value changed function: YSourceCheckBox
        function YSourceCheckBoxValueChanged(app, event)

        end

        % Value changed function: PMomentumFluxCheckBox
        function PMomentumFluxCheckBoxValueChanged(app, event)

        end

        % Value changed function: QmomentumFluxCheckBox
        function QmomentumFluxCheckBoxValueChanged(app, event)

        end

        % Value changed function: FxFluxCheckBox
        function FxFluxCheckBoxValueChanged(app, event)

        end

        % Value changed function: FyFluxCheckBox
        function FyFluxCheckBoxValueChanged(app, event)

        end

        % Value changed function: GxFluxCheckBox
        function GxFluxCheckBoxValueChanged(app, event)

        end

        % Value changed function: GyFluxCheckBox
        function GyFluxCheckBoxValueChanged(app, event)

        end

        % Value changed function: BreakingAgeCheckBox
        function BreakingAgeCheckBoxValueChanged(app, event)

        end

        % Value changed function: MaximumWaveHeightCheckBox
        function MaximumWaveHeightCheckBoxValueChanged(app, event)

        end

        % Value changed function: MinimumWaveHeightCheckBox
        function MinimumWaveHeightCheckBoxValueChanged(app, event)

        end

        % Value changed function: MaximumVelocityCheckBox
        function MaximumVelocityCheckBoxValueChanged(app, event)

        end

        % Value changed function: UndertowCheckBox
        function UndertowCheckBoxValueChanged(app, event)

        end

        % Value changed function: MaxVorticityCheckBox
        function MaxVorticityCheckBoxValueChanged(app, event)

        end

        % Value changed function: MaxMomentumFluxCheckBox
        function MaxMomentumFluxCheckBoxValueChanged(app, event)

        end

        % Value changed function: ArrivalCheckBox
        function ArrivalCheckBoxValueChanged(app, event)
        if app.ArrivalCheckBox.Value
            app.ThresholdEditFieldLabel.Visible = "on";
            app.ThresholdEditField.Visible = "on";
        else
            app.ThresholdEditFieldLabel.Visible = "off";
            app.ThresholdEditField.Visible = "off";
            app.ThresholdEditField.FontColor = '0.65,0.65,0.65';
        end


        end

        % Value changed function: WaveHeightCheckBox
        function WaveHeightCheckBoxValueChanged(app, event)

        end

        % Value changed function: RollerinducedFluxCheckBox
        function RollerinducedFluxCheckBoxValueChanged(app, event)

        end

        % Value changed function: PressureFieldCheckBox
        function PressureFieldCheckBoxValueChanged(app, event)

        end

        % Value changed function: BreakingLocationCheckBox
        function BreakingLocationCheckBoxValueChanged(app, event)

        end

        % Callback function
        function EastlimitmEditFieldValueChanged(app, event)
            app.EastEditField.FontColor = 'k';
        end

        % Callback function
        function WestbountmEditFieldValueChanged(app, event)
            app.WestEditField.FontColor = 'k';
        end

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

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

        % Callback function
        function ButtonGroup_5SelectionChanged(app, event)

        end

        % Callback function
        function DirectSpongeSwitchValueChanged(app, event)

             if strcmp(app.DirectSpongeSwitch.Value,'On')
                app.DecayTypeDropDown.Visible = "on";
                app.DecayTypeDropDownLabel.Visible = "on";
                app.Panel_16.Enable = "on";
             elseif strcmp(app.DirectSpongeSwitch.Value,'Off')
                app.DecayTypeDropDown.Visible = "off";
                app.DecayTypeDropDownLabel.Visible = "off";
             end



        end

        % Callback function
        function Button_9Pushed(app, event)
               maindir = uigetdir;
               maindir = fullfile(maindir,' ');
               app.ResultsfolderEditField.Value = deblank(string(maindir));
               app.ResultsfolderEditField.FontColor = 'k';
        end

        % Button pushed function: Button_7
        function Button_7Pushed2(app, event)
            [filename3,path3] = uigetfile('*.txt;*.shp');
            fullname = fullfile(path3,filename3);
            try
            [~, name, ~] = fileparts(filename3);
            app.GaugeFname = name;
            app.LoadFileEditField.Value = filename3;
            app.GAUGEFILE = filename3;
            app.GAUGEFILElocation = fullname;
            app.LoadFileEditField.FontColor = 'k';
            app.LoadFileEditField.BackgroundColor = 'w';

            if contains(filename3,'.shp')
                gauge = shaperead(fullname);
                latGT = [];
                lonGT = [];
                app.latGauge = latGT;

                for k = 1:length(gauge)
```

```
                    latGT = [latGT, gauge(k).X];  % Concatenate X coordinates
                    lonGT = [lonGT, gauge(k).Y];  % Concatenate Y coordinates
                end

                list = [round(lonGT,4)',round(latGT,4)'];
                list =  unique(list,'rows'); %remove duplicate lat-long combination
                app.NumberStations = num2str(height(list));
                app.STATION_FILE = string(name) + ".txt";

                %Save as text file
                app.GaugeListGrids = list;

            elseif contains(filename3,'.txt')
                gaugetext = readmatrix(fullname);
                app.gaugetext0 = gaugetext;
                app.NumberStations = num2str(height(gaugetext));
                app.STATION_FILE = string(filename3);

            end

            catch
                %do nothing
            end

        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: ResultsfolderEditField
        function ResultsfolderEditFieldValueChanged(app, event)


        end

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

        end

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

        end

        % Value changed function: TotalEditField
        function TotalEditFieldValueChanged2(app, event)
            app.TotalEditField.FontColor = 'k';
```

```matlab

        end

        % Button pushed function: Button_32
        function Button_4Pushed(app, event)
                [bathymetryfile,path2] = uigetfile('*.*');
                app.SurfaceHeightEditField.Value = string(bathymetryfile);
                app.SurfaceHeightEditField.FontColor = 'k';
                app.SurfaceHeightEditField.BackgroundColor = 'w';
                app.SurfaceHeightLOC = string(path2) + string(bathymetryfile);
                app.SurfaceHeightNAME = string(bathymetryfile);


        end

        % Button pushed function: Button_34
        function Button_5Pushed(app, event)
            [file,path2] = uigetfile('*.*');
                app.UVelocityEditField.Value = string(file);
                app.UVelocityEditField.FontColor = 'k';
                app.UVelocityEditField.BackgroundColor = 'w';
                app.XVelocityLOC = string(path2) + string(file);
                app.XVelocityNAME = string(file);

        end

        % Button pushed function: Button_33
        function Button_6Pushed(app, event)
                [file,path2] = uigetfile('*.*');
                app.VVelocityEditField.Value = string(file);
                app.VVelocityEditField.FontColor = 'k';
                app.VVelocityEditField.BackgroundColor = 'w';
                app.YVelocityLOC = string(path2) + string(file);
                app.YVelocityNAME = string(file);
        end

        % Selection changed function: ButtonGroup_18
        function ButtonGroup_6SelectionChanged(app, event)

                   if  app.ImportUVZButton.Value
                       app.TabGroup2.SelectedTab =   app.Tab;

                   end

                    if app.WavemakerButton.Value
                       app.TabGroup2.SelectedTab = app.Tab_2;

                    end

                   if app.MeteotsunamiButton.Value
                        app.TabGroup2.SelectedTab = app.Tab_4;
                        app.MinimumWaveHeightCheckBox.Value = 1;
                        app.PressureFieldCheckBox.Value = 1;
                   else
                       %uncheck the expected output values
                        app.MinimumWaveHeightCheckBox.Value = 0;
                        app.PressureFieldCheckBox.Value = 0;
                   end


                   if app.VesselButton.Value
                       app.TabGroup2.SelectedTab = app.Tab_6;

                   end

        end

        % Value changed function: DepthFileEditField
        function DepthFileEditFieldValueChanged(app, event)
          app.DepthFileEditField.BackgroundColor = 'w';
          app.DepthFileEditField.FontColor = 'k';

        end

        % Selection changed function: ButtonGroup_7
        function ButtonGroup_7SelectionChanged(app, event)
            %List all the items to be displayed or hidden
            function setVisibility(app, depthVisible, angleVisible, startXVisible, button8Visible, depthFileVisible, waterLevelVisible, invertValuesVisible)
                app.DepthmEditFieldLabel.Visible = depthVisible;
                app.DepthmEditField.Visible = depthVisible;
                app.AngleEditFieldLabel.Visible = angleVisible;
                app.AngleEditField.Visible = angleVisible;
                app.StartatXEditFieldLabel.Visible = startXVisible;
                app.StartatXEditField.Visible = startXVisible;
                app.Button_8.Visible = button8Visible;
                app.DepthFileEditField.Visible = depthFileVisible;
                app.WaterlevelmEditFieldLabel.Visible = waterLevelVisible;
                app.WaterlevelmEditField.Visible = waterLevelVisible;
                app.InvertvaluesCheckBox.Visible = invertValuesVisible;
            end


            if app.ImportDataButton.Value
                setVisibility(app, "off", "off", "off", "on", "on", "on", "on");
            elseif app.FlatButton.Value
                setVisibility(app, "on", "off", "off", "off", "off", "off", "off");
            elseif app.SlopeButton.Value
                setVisibility(app, "on", "on", "on", "off", "off", "off", "off");
            end

            % Set font color to black
            if ~app.Button4.Value
                app.ImportDataButton.FontColor = 'k';
                app.SlopeButton.FontColor = 'k';
                app.FlatButton.FontColor = 'k';
            end

        end

        % Size changed function: InputTab
        function InputTabSizeChanged(app, event)

        end

        % Value changed function: DepthmEditField
        function DepthmEditFieldValueChanged(app, event)

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

        end

        % Value changed function: AngleEditField
        function AngleEditFieldValueChanged(app, event)
            app.AngleEditField.FontColor = 'k';
            app.AngleEditField.BackgroundColor = 'w';
        end

        % Value changed function: StartatXEditField
        function StartatXEditFieldValueChanged(app, event)
           app.StartatXEditField.FontColor = 'k';
           app.StartatXEditField.BackgroundColor = 'w';

        end

        % Value changing function: ResultsfolderEditField
        function ResultsfolderEditFieldValueChanging(app, event)

            app.ResultsfolderEditField.BackgroundColor = 'w';
            app.ResultsfolderEditField.FontColor = 'k';
        end

        % Value changed function: YEditField_2
        function YEditField_2ValueChanged(app, event)
           app.YEditField_2.FontColor = 'k';

        end

        % Button pushed function: Button_10
        function Button_10Pushed(app, event)
               [frictionfile,frictionpath] = uigetfile('*.txt;*.tif');
               app.FrictionFileEditField.Value = string(frictionfile);
               app.Cd_file = fullfile(string(frictionpath) + string(frictionfile));
               app.filenamefriction = string(frictionfile);
               app.FrictionFileEditField.FontColor = 'k';
               app.FrictionFileEditField.BackgroundColor = 'w';
        end

        % Value changing function: FrictionFileEditField
        function FrictionFileEditFieldValueChanging(app, event)
                app.FrictionFileEditField.Value = '';
                app.FrictionFileEditField.FontColor = 'k';
                app.FrictionFileEditField.BackgroundColor = 'w';
        end

        % Selection changed function: ButtonGroup_8
        function ButtonGroup_8SelectionChanged(app, event)
            if app.ConstantCoefficientButton.Value
```

```
                app.FrictionFileEditField.Visible = "off";
                app.FrictionFileEditFieldLabel.Visible = "off";
                app.Button_10.Visible = "off";
                app.ConstantFrictionValue.Visible = "on";
                app.ValueEditFieldLabel.Visible = "on";
%                 app.BottomFrictionEditField.Enable = "on";
%                 app.BottomFrictionEditFieldLabel.Enable  = "on";
            elseif app.NonconstantButton.Value
                app.FrictionFileEditField.Visible = "on";
                app.FrictionFileEditFieldLabel.Visible = "on";
                app.Button_10.Visible = "on";
                app.ConstantFrictionValue.Visible = "off";
                app.ValueEditFieldLabel.Visible = "off";
%                 app.BottomFrictionEditField.Enable = "on";
%                 app.BottomFrictionEditFieldLabel.Enable  = "on";
            end

        end

        % Value changed function: ConstantFrictionValue
        function ConstantFrictionValueValueChanged(app, event)
            app.ConstantFrictionValue.FontColor = 'k';
            app.ConstantFrictionValue.BackgroundColor = 'w';
        end

        % Value changed function: C2EditField
        function C2EditFieldValueChanged(app, event)
             app.C2EditField.FontColor = 'k';


        end

        % Value changed function: C1EditField
        function C1EditFieldValueChanged(app, event)

             app.C1EditField.FontColor = 'k';
        end

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


        end

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

        end

        % Value changing function: SavefilestoEditField
        function SavefilestoEditFieldValueChanging(app, event)
            app.SavefilestoEditField.Value = '';
            app.SavefilestoEditField.FontColor = 'k';
            app.SavefilestoEditField.BackgroundColor = 'w';

        end

        % Value changed function: DecayTypeDropDown
        function DecayTypeDropDownValueChanged(app, event)

        end

        % Button pushed function: ObstacleLoadFile
        function Button_19Pushed(app, event)
             [filename,path] = uigetfile('*.txt;*.tif');
             app.ObstacleFilename.Value = string(filename);
             app.ObstacleFilename.FontColor = 'k';
             app.ObstacleFilename.BackgroundColor = 'w';
             app.obstacleLOC = string(path) + string(filename);
             app.obstacleNAME = string(filename);
        end

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

        end

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

        end

        % Value changed function: WavemakertypeDropDown
        function WavemakertypeDropDownValueChanged(app, event)
%Disable and reset all fields and buttons first
allFields = [app.XcoordinatemEditField, app.YcoordinatemEditField, app.WidthmEditField, app.WidthDeltaEditField, app.AmplitudemEditField, app.WaterdepthmEditField, app.LagtimesecEditField, app.ThetadegreesEditField, app.TimerampsecEditField, app.PeriodsecEditField, app.WavecomponentfileEditField, app.WaveCompCountEditField,app.MinimumEditField,app.MaximumEditField,app.PeakEditField, app.WaveHeightmEditField];
for field = allFields
    field.Enable = 'off';
    field.FontColor = [0.8, 0.8, 0.8];
    if isprop(field, 'Value') && isnumeric(field.Value)
        field.Value = 0;
    elseif isprop(field, 'Value')
        field.Value = "";
    end
end
app.Button_20.Enable = 'off'; %Disable import buttom of 'Save Comoponent File'


%List of the paremeters to be enabled based on the wavemaker type selected
wavemakerSettings = struct(...
    'INI_REC', {{app.XcoordinatemEditField, app.YcoordinatemEditField, app.WidthmEditField}}, ...
    'LEF_SOL', {{app.AmplitudemEditField, app.WaterdepthmEditField, app.LagtimesecEditField}}, ...
    'INI_SOL', {{app.AmplitudemEditField, app.WaterdepthmEditField, app.XcoordinatemEditField}}, ...
    'WK_IRR', {{app.XcoordinatemEditField,app.YcoordinatemEditField,app.
WidthmEditField,app.TimerampsecEditField,app.WidthDeltaEditField,app.MinimumEditField,app.MaximumEditField,app.PeakEditField,app.WaveHeightmEditField,app.WaterdepthmEditField}}, ...
    'JON_1D', {{app.XcoordinatemEditField,app.YcoordinatemEditField,app.WidthmEditField,app.TimerampsecEditField,app.WidthDeltaEditField,app.MinimumEditField,app.MaximumEditField,app.PeakEditField,app.WaveHeightmEditField,app.WaterdepthmEditField}}, ...
    'JON_2D', {{app.XcoordinatemEditField,app.YcoordinatemEditField,app.WidthmEditField,app.TimerampsecEditField,app.WidthDeltaEditField,app.MinimumEditField,app.MaximumEditField,app.PeakEditField,app.WaveHeightmEditField,app.WaterdepthmEditField}}, ...
    'TMA_1D', {{app.XcoordinatemEditField,app.YcoordinatemEditField,app.WidthmEditField,app.TimerampsecEditField,app.WidthDeltaEditField,app.MinimumEditField,app.MaximumEditField,app.PeakEditField,app.WaveHeightmEditField,app.WaterdepthmEditField}}, ...
    'WK_REG', {{app.XcoordinatemEditField, app.AmplitudemEditField, app.WaterdepthmEditField, app.ThetadegreesEditField, app.TimerampsecEditField,app.WidthDeltaEditField}}, ...
    'WK_TIME_SERIES', {{app.WavecomponentfileEditField, app.Button_20, app.PeriodsecEditField, app.WaterdepthmEditField, app.XcoordinatemEditField, app.WidthmEditField, app.WaveCompCountEditField}}, ...
    'WK_DATA2D', {{app.XcoordinatemEditField, app.YcoordinatemEditField, app.WidthDeltaEditField, app.WaterdepthmEditField, app.WavecomponentfileEditField, app.Button_20}}, ...
    'INI_GAUSSIAN', {{app.AmplitudemEditField, app.XcoordinatemEditField, app.YcoordinatemEditField, app.WidthmEditField}} ...
);

%Enable paremeters based on the wavemaker type selected
selectedType = app.WavemakertypeDropDown.Value;
if isfield(wavemakerSettings, selectedType)
    fieldsToEnable = wavemakerSettings.(selectedType);
    for fieldCell = fieldsToEnable
        field = fieldCell{:};
        field.Enable = 'on';
    end
end

if ~strcmp(app.WavemakertypeDropDown.Value, 'WK_TIME_SERIES')
    app.WavecomponentfileEditField.BackgroundColor = 'w';
    app.WavecomponentfileEditField.Value = '';
    app.WavecomponentfileEditField.FontColor = 'k';
end

        end

        % Drop down opening function: WavemakertypeDropDown
        function WavemakertypeDropDownOpening(app, event)
            app.WavemakertypeDropDown.FontColor = '0.00,0.00,0.00';
        end

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

        % Value changed function: WaterdepthmEditField
        function WaterdepthmEditFieldValueChanged(app, event)
```

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


        end

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

        end

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


        end

        % Callback function
        function XcentercoordinateEditFieldValueChanged(app, event)


        end

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


        end

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


        end

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


        end

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


        end

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


        end
```

```matlab

        % Button pushed function: Button_20
        function Button_20Pushed(app, event)
              [file,path] = uigetfile('*.*'); %*.txt;
                app.WavecomponentfileEditField.Value = string(file);
                app.WaveFileNAME = string(file);
                app.WaveFileLOC = string(path) + string(file);
                app.WavecomponentfileEditField.BackgroundColor = 'w';
                app.WavecomponentfileEditField.FontColor =  'k';

        end

        % Value changed function: WavecomponentfileEditField
        function WavecomponentfileEditFieldValueChanged(app, event)


        end

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

        end

        % Value changed function: XEditField_2
        function XEditField_2ValueChanged(app, event)
           app.XEditField_2.FontColor = 'k';


        end

        % Callback function
        function TimeintervalsecEditFieldValueChanged(app, event)
            app.TimeintervalsecEditField.FontColor = 'k';

        end

        % Callback function
        function SteadyTimeEditFieldValueChanged(app, event)
            app.SteadyTimeEditField.FontColor = 'k';

        end

        % Callback function
        function ButtonGroup_12SelectionChanged(app, event)


        end

        % Callback function
        function SchemeDropDownValueChanged(app, event)

        end

        % Value changing function: UVelocityEditField
        function UVelocityEditFieldValueChanging(app, event)

        end

        % Value changing function: SurfaceHeightEditField
        function SurfaceHeightEditFieldValueChanging(app, event)


        end

        % Callback function
        function Button5Pushed(app, event)
            app.TabGroup2.SelectedTab =   app.Tab_3;        %app.InputsTab;
        end

        % Value changing function: VVelocityEditField
        function VVelocityEditFieldValueChanging(app, event)


        end

        % Value changed function: WindforceSwitch
        function WindforceSwitchValueChanged(app, event)
              if strcmp(app.WindforceSwitch.Value,"On")
                  app.WindwaveinteractionSwitch.Enable = "on";
                  app.WindwaveinteractionSwitchLabel.Enable = "on";
                  app.WindstresscoefficientEditField.Enable = "on";
                  app.WindstresscoefficientEditFieldLabel.Enable = "on";
                  app.CrestRatioEditField.Enable = "on";
                  app.CrestRatioEditFieldLabel.Enable ="on";

              elseif strcmp(app.WindforceSwitch.Value,"Off")
                  app.WindwaveinteractionSwitch.Enable = "off";
                  app.WindwaveinteractionSwitchLabel.Enable = "off";
                  app.WindstresscoefficientEditField.Enable = "off";
                  app.WindstresscoefficientEditFieldLabel.Enable = "off";
                  app.CrestRatioEditField.Enable = "off";
                  app.CrestRatioEditFieldLabel.Enable ="off";
                  app.WindstresscoefficientEditField.FontColor = '0.65,0.65,0.65';
                  app.WindstresscoefficientEditField.FontColor = '0.65,0.65,0.65';
                  app.CrestRatioEditField.FontColor = '0.65,0.65,0.65';
              end

        end

        % Value changed function: ConstantwindfieldSwitch
        function ConstantwindfieldSwitchValueChanged(app, event)

            if strcmp(app.ConstantwindfieldSwitch.Value,"On")
                app.ConstantwindfileEditFieldLabel.Enable = "on";
                app.ConstantwindfileEditField.Enable = "on";
                app.Button_29.Enable = "on";
                app.WindforceSwitch.Value = "On";

                app.ConstantwindfieldSwitch.Enable = "on";
                app.ConstantwindfieldSwitchLabel.Enable = "on";
                app.WindwaveinteractionSwitch.Enable = "on";
                app.WindwaveinteractionSwitchLabel.Enable = "on";
                app.WindstresscoefficientEditField.Enable = "on";
                app.WindstresscoefficientEditFieldLabel.Enable = "on";
                app.CrestRatioEditField.Enable = "on";
                app.CrestRatioEditFieldLabel.Enable ="on";

            elseif strcmp(app.ConstantwindfieldSwitch.Value,"Off")
                app.ConstantwindfileEditFieldLabel.Enable = "off";
                app.ConstantwindfileEditField.Enable = "off";
                app.Button_29.Enable = "off";
                app.WindforceSwitch.Value = "Off";
                app.WindwaveinteractionSwitch.Enable = "off";
                app.WindwaveinteractionSwitchLabel.Enable = "off";
                app.WindstresscoefficientEditField.Enable = "off";
                app.WindstresscoefficientEditFieldLabel.Enable = "off";
                app.CrestRatioEditField.Enable = "off";
                app.CrestRatioEditFieldLabel.Enable ="off";
            end

        end

        % Button pushed function: Button_29
        function Button_29Pushed(app, event)
            [file,path2] = uigetfile('*.*');
            app.ConstantwindfileEditField.Value = string(file);
            app.ConstantwindfileEditField.FontColor = 'k';
            app.ConstantwindfileEditField.BackgroundColor = 'w';
                app.ConsWindLOC = string(path2) + string(file);
                app.ConsWindNAME = string(file);
        end

        % Value changed function: CrestRatioEditField
        function CrestRatioEditFieldValueChanged(app, event)

        end

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

        end

        % Button pushed function: Button_30
        function Button_30Pushed(app, event)
              [file,path2] = uigetfile('*.*');
                app.StormfileEditField.Value = string(file);
                app.StormfileEditField.FontColor = 'k';
                app.StormfileEditField.BackgroundColor = 'w';
                app.StormFileLOC = string(path2) + string(file);
                app.StormFileNAME = string(file);
        end

        % Callback function
        function Button_39Pushed(app, event)
               [file,path2] = uigetfile('*.*');
                app.PressurefileEditField.Value = string(file);
                app.PressurefileEditField.FontColor = 'k';
                app.PressurefileEditField.BackgroundColor = 'w';
                app.PressureFileLOC = string(path2) + string(file);
                app.PressureFileNAME = string(file);
        end

        % Value changed function: WindwaveinteractionSwitch
        function WindwaveinteractionSwitchValueChanged(app, event)


        end

        % Value changed function: AirpressureSwitch
        function AirpressureSwitchValueChanged(app, event)
            if strcmp(app.AirpressureSwitch.Value,"On")
                app.PressurefileEditField.Enable = "on";
                app.PressurefileEditFieldLabel.Enable = "on";
                app.Button_43.Enable = "on";
            else
                app.PressurefileEditField.Enable = "off";
                app.PressurefileEditFieldLabel.Enable = "off";
                app.Button_43.Enable = "off";
            end
        end

        % Value changed function: HollandstormmodelSwitch
        function HollandstormmodelSwitchValueChanged(app, event)
            if strcmp(app.HollandstormmodelSwitch.Value,"On")
                app.StormfileEditFieldLabel.Enable = "on";
                app.StormfileEditField.Enable = "on";
                app.Button_30.Enable = "on";
            else
                app.StormfileEditFieldLabel.Enable = "off";
                app.StormfileEditField.Enable = "off";
                app.Button_30.Enable = "off";
            end
        end

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

        end

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

        end

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

        end

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

        end

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

        end

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

        end

        % Value changed function: RungeKuttaparameter1EditField
        function RungeKuttaparameter1EditFieldValueChanged(app, event)
            app.RungeKuttaparameter1EditField.FontColor = 'k';

        end

        % Value changed function: RungeKuttaparameter2EditField
        function RungeKuttaparameter2EditFieldValueChanged(app, event)
            app.RungeKuttaparameter2EditField.FontColor = 'k';

        end

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

        end

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

        end

        % Callback function
        function PressurefileEditFieldValueChanged(app, event)
            app.PressurefileEditField.FontColor = 'k';

        end

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

        end

        % Value changed function: RungeKuttaparam2EditField_2
        function RungeKuttaparam2EditField_2ValueChanged(app, event)
           app.RungeKuttaparam2EditField_2.FontColor = 'k';

        end

        % Button pushed function: Button_35
        function Button_35Pushed(app, event)
            [file,path2] = uigetfile('*.*');
            app.FileEditField.Value = string(file);
```

```
                app.FileEditField.FontColor = 'k';
                app.FileEditField.BackgroundColor = 'w';
                app.HardBottomFileLOC = string(path2) + string(file);
                app.HardBottomFileNAME = string(file);
        end

        % Selection changed function: ButtonGroup_21
        function ButtonGroup_21SelectionChanged(app, event)
           if app.HardbednoerosionButton.Value
               app.Button_35.Visible= "on";
               app.FileEditField.Visible = "on";
               app.FileEditFieldLabel.Visible = "on";
           elseif app.SedimentbedButton.Value

               app.Button_35.Visible = "off";
               app.FileEditField.Visible = "off";
               app.FileEditFieldLabel.Visible = "off";
           end

           if ~app.Button_37.Value
               app.SedimentbedButton.FontColor = 'k';
               app.HardbednoerosionButton.FontColor = 'k';
           end

        end

        % Selection changed function: ButtonGroup_20
        function ButtonGroup_20SelectionChanged(app, event)
            if ~app.Button_36.Value
                app.NobedchangeButton.FontColor = 'k';
                app.UpdatedepthButton.FontColor = 'k';
            end
        end

        % Selection changed function: ButtonGroup_19
        function ButtonGroup_19SelectionChanged(app, event)

        end

        % Selection changed function: ButtonGroup_22
        function ButtonGroup_22SelectionChanged(app, event)
            if app.YesButton_8.Value
                 app.RungeKuttaparam2EditField_2.Enable = "on";
                 app.RungeKuttaparam2EditFieldLabel_2.Enable = "on";
            elseif app.NoButton_7.Value
                app.RungeKuttaparam2EditField_2.Enable = "off";
                app.RungeKuttaparam2EditFieldLabel_2.Enable = "off";
            end


        end

        % Button pushed function: Button_40
        function Button_40Pushed(app, event)

            [files, path] = uigetfile('*.*','Select the files','MultiSelect',↙
'on');
              app.ImportvesselfilesTextArea.Value = string(files);
              app.VesselLOC = string(path) + string(files);
              app.VesselNAME = string(files); %string(path);
              app.ImportvesselfilesTextArea.FontColor = 'k';
              app.ImportvesselfilesTextArea.BackgroundColor = 'w';

              if ischar(files)
                  % Only one file was selected
                  app.VesselNumber = 1;
              elseif iscell(files)
                  % Multiple files were selected
                  app.VesselNumber = numel(files);
              end


        end

        % Value changed function: FileEditField
        function FileEditFieldValueChanged(app, event)
              [file,path] = uigetfile('*.*');
              app.FileEditField.Value = string(file);
              app.FileEditField.FontColor = 'k';
              app.FileEditField.BackgroundColor = 'w';
              app.HardbottomLOC = string(path) + string(file);
              app.HardbottomtNAME = string(file);
        end

        % Value changed function: DirectCheckBox
        function DirectCheckBoxValueChanged(app, event)
if app.DirectCheckBox.Value
    app.DecayTypeDropDown.Visible = "on";
    app.DecayTypeDropDownLabel.Visible = "on";
else
    app.DecayTypeDropDown.Visible = "off";
    app.DecayTypeDropDownLabel.Visible = "off";
end


if app.DirectCheckBox.Value || app.FrictionCheckBox.Value || app.DiffusionCheckBox.Value
    %if there are other buttons checked in Sponge Boundary, the N,E,W,S textboxes are not disabled
    app.WestEditField.Enable = "on";
    app.SouthEditField.Enable = "on";
    app.EastEditField.Enable = "on";
    app.NorthEditField.Enable = "on";
else
    %Include disable the N,E,W,S textboxes
    app.DecayTypeDropDown.Visible = "off";
    app.DecayTypeDropDownLabel.Visible = "off";
    app.WestEditField.Enable = "off";
    app.SouthEditField.Enable = "off";
    app.EastEditField.Enable = "off";
    app.NorthEditField.Enable = "off";
end


        end

        % Value changed function: FrictionCheckBox
        function FrictionCheckBoxValueChanged(app, event)
            if app.FrictionCheckBox.Value
                app.MaximumCdEditFieldLabel.Visible = "on";
                app.MaximumCdEditField.Visible = "on";
            else
                app.MaximumCdEditField.Visible = "off";
                app.MaximumCdEditFieldLabel.Visible = "off";
            end

            if app.DirectCheckBox.Value || app.FrictionCheckBox.Value || app.DiffusionCheckBox.Value
                %if there are other buttons checked in Sponge Boundary, the N,E,W,S textboxes are not disabled
                app.WestEditField.Enable = "on";
                app.SouthEditField.Enable = "on";
                app.EastEditField.Enable = "on";
                app.NorthEditField.Enable = "on";
            else
                %Include disable the N,E,W,S textboxes
                app.DecayTypeDropDown.Visible = "off";
                app.DecayTypeDropDownLabel.Visible = "off";
                app.WestEditField.Enable = "off";
                app.SouthEditField.Enable = "off";
                app.EastEditField.Enable = "off";
                app.NorthEditField.Enable = "off";
            end

        end

        % Value changed function: DiffusionCheckBox
        function DiffusionCheckBoxValueChanged(app, event)
            if app.DiffusionCheckBox.Value
                app.MaximumCspEditField.Visible = "on";
                app.MaximumCspEditFieldLabel.Visible = "on";
            else
                app.MaximumCspEditFieldLabel.Visible = "off";
                app.MaximumCspEditField.Visible = "off";
            end

            if app.DirectCheckBox.Value || app.FrictionCheckBox.Value || app.DiffusionCheckBox.Value
                %if there are other buttons checked in Sponge Boundary, the N,E,W,S textboxes are not disabled
                app.WestEditField.Enable = "on";
                app.SouthEditField.Enable = "on";
                app.EastEditField.Enable = "on";
                app.NorthEditField.Enable = "on";
            else
                %Include disable the N,E,W,S textboxes
                app.DecayTypeDropDown.Visible = "off";
                app.DecayTypeDropDownLabel.Visible = "off";
                app.WestEditField.Enable = "off";
                app.SouthEditField.Enable = "off";
                app.EastEditField.Enable = "off";
                app.NorthEditField.Enable = "off";
            end

        end

        % Callback function
        function UseBreakingIndexCheckBoxValueChanged(app, event)

            if app.UseBreakingIndexCheckBox.Value
                app.C1EditField.Visible = "on";
                app.C1EditFieldLabel.Visible = "on";
                app.C2EditField.Visible ="on";
                app.C2EditFieldLabel.Visible = "on";
                app.SchemeDropDown.Visible = "on";
                app.SchemeDropDownLabel.Visible = "on";
            else
                app.C1EditField.Visible = "off";
                app.C1EditFieldLabel.Visible = "off";
                app.C2EditField.Visible ="off";
                app.C2EditFieldLabel.Visible = "off";
                app.SchemeDropDown.Visible = "off";
                app.SchemeDropDownLabel.Visible = "off";
            end
        end

        % Callback function
        function UseSmagorinskymixingCheckBoxValueChanged(app, event)
            if app.UseSmagorinskyCheckBox.Value
                app.TimeintervalsecEditField.Visible = "on";
                app.TimeintervalsecEditFieldLabel.Visible = "on";
                app.SteadyTimeEditField.Visible = "on";
                app.SteadyTimeEditFieldLabel.Visible = "on";
            else
                app.TimeintervalsecEditField.Visible = "off";
                app.TimeintervalsecEditFieldLabel.Visible = "off";
                app.SteadyTimeEditField.Visible = "off";
                app.SteadyTimeEditFieldLabel.Visible = "off";
            end
        end

        % Value changed function: AddobstaclesCheckBox
        function AddobstaclesCheckBoxValueChanged(app, event)
          if app.AddobstaclesCheckBox.Value
                app.ObstacleFilename.Visible ="on";
                app.ObstacleLoadFile.Visible = "on";
          else
                app.ObstacleFilename.Visible ="off";
                app.ObstacleLoadFile.Visible = "off";
          end


        end

        % Value changed function: CreatenestedgridboundaryCheckBox
        function CreatenestedgridboundaryCheckBoxValueChanged(app, event)
            %Define the fields to be enabled/disabled under the nested grid option
            fieldsToToggle = {
                app.SpacingEditFieldLabel, app.SpacingEditField, app.WestEditField_2, ...
                app.SouthEditField_2, app.EastEditField_2, app.
```

```
NorthEditField_2, app.BoundarylimitsButton
                };

            if app.CreatenestedgridboundaryCheckBox.Value
                % Disable the items under the 'Import gauge list' option
                app.ImportgaugelistCheckBox.Value = false;
                app.LoadFileEditField.Enable = "off";
                app.Button_7.Enable = "off";

                % Enable fields under the nested grid option
                for i = 1:length(fieldsToToggle)
                    fieldsToToggle{i}.Enable = "on";
                end
            else
                % Set font color to gray
                grayColor = '0.65,0.65,0.65';
                app.SpacingEditField.FontColor = grayColor;
                app.WestEditField_2.FontColor = grayColor;
                app.SouthEditField_2.FontColor = grayColor;
                app.EastEditField_2.FontColor = grayColor;
                app.NorthEditField_2.FontColor = grayColor;

                %Disable fields under the nested grid option
                for i = 1:length(fieldsToToggle)
                    fieldsToToggle{i}.Enable = "off";
                end
            end

        end

        % Value changing function: DepthFileEditField
        function DepthFileEditFieldValueChanging(app, event)


        end

        % Value changed function: ImportgaugelistCheckBox
        function ImportgaugelistCheckBoxValueChanged(app, event)
            app.WestEditField_2.Enable = "off";
            app.SouthEditField_2.Enable = "off";
            app.EastEditField_2.Enable = "off";
            app.NorthEditField_2.Enable = "off";
            app.BoundarylimitsButton.Enable = "off";

            if app.ImportgaugelistCheckBox.Value
                app.CreatenestedgridboundaryCheckBox.Enable = "on";
                app.LoadFileEditField.Enable = "on";
                app.Button_7.Enable = "on";
                app.CreatenestedgridboundaryCheckBox.Value = false;
                app.SpacingEditFieldLabel.Enable = "off";
                app.SpacingEditField.Enable = "off";
                app.SpacingEditFieldLabel.Enable = 'off';

            else
                app.LoadFileEditField.Enable = "off";
                app.Button_7.Enable = "off";  %To upload gauge data
                app.SpacingEditFieldLabel.Enable = "off";
                app.SpacingEditField.Enable = "off";
                app.SpacingEditFieldLabel.Enable = "off";


            end
        end

        % Value changed function: LoadFileEditField
        function LoadFileEditFieldValueChanged(app, event)


        end

        % Value changing function: LoadFileEditField
        function LoadFileEditFieldValueChanging(app, event)


        end

        % Selection change function: TabGroup2
        function TabGroup2SelectionChanged(app, event)
            selectedTab = app.TabGroup2.SelectedTab;

        end

        % Value changed function: UseSmagorinskyCheckBox
        function UseSmagorinskyCheckBoxValueChanged2(app, event)
           if app.UseSmagorinskyCheckBox.Value
           app.TimeintervalsecEditFieldLabel.Visible = "on";
           app.TimeintervalsecEditField.Visible = "on";
           app.SteadyTimeEditFieldLabel.Visible = "on";
           app.SteadyTimeEditField.Visible = "on";
        else
           app.TimeintervalsecEditFieldLabel.Visible = "off";
           app.TimeintervalsecEditField.Visible = "off";
           app.SteadyTimeEditFieldLabel.Visible = "off";
           app.SteadyTimeEditField.Visible = "off";
           end
        end

        % Value changed function: SavefilestoEditField
        function SavefilestoEditFieldValueChanged(app, event)


        end

        % Button pushed function: Button_41
        function Button_41Pushed(app, event)
               workingfolder = uigetdir;
               workingfolder = fullfile(workingfolder,' ');
               app.Mpif90EditField.Value = deblank(string(workingfolder));
               app.Mpif90EditField.FontColor = 'k';
        end

        % Callback function
        function Button_42Pushed(app, event)
            workingfolder = uigetdir;
            workingfolder = fullfile(workingfolder,' ');
            app.MakefileEditField.Value = deblank(string(workingfolder));
            app.MakefileEditField.FontColor = 'k';
            app.MakefileEditField.BackgroundColor = 'w';
        end

        % Callback function
        function IncludefunwaveexeinthegenerationCheckBoxValueChanged(app, event)

        end

        % Callback function
        function MakefileEditFieldValueChanging(app, event)
            app.MakefileEditField.FontColor = 'k';
            app.MakefileEditField.BackgroundColor = 'w';
        end

        % Value changed function: Mpif90EditField
        function Mpif90EditFieldValueChanged(app, event)


        end

        % Value changing function: Mpif90EditField
        function Mpif90EditFieldValueChanging(app, event)
            app.Mpif90EditField.FontColor = 'k';
            app.Mpif90EditField.BackgroundColor = 'w';
        end

        % Button pushed function: StartsimulationButton
        function StartsimulationButtonPushed(app, event)
            %Show the To disable the button whilst the rest of the code is running
            app.TabGroup3.SelectedTab = app.LogReportTab;
            % app.StartsimulationButton.Enable = 'off';
            drawnow;

            cd(fullfile(app.directory));


            if exist(app.ResultsfolderEditField.Value, 'dir')
                rmdir(app.ResultsfolderEditField.Value, 's'); % Delete the pre-existing folder and its contents
            end

            LogFolder = fullfile(app.directory, 'Log_Files');

            % Check if the 'INPUTFOLDER' directory exists, create it if it doesn't
            if ~exist(LogFolder, 'dir')
                mkdir(LogFolder);  % Creates the 'INPUTFOLDER' directory
            end

            logFilePath = fullfile(LogFolder,'Log_Report_InputPrep.txt'); % Define the log file path correctly
            currentLogContent = fileread(logFilePath); % Read the current content of the log file

            app.Preview_LogReport.Value = fileread(logFilePath);
            drawnow;

            % Ensure the current directory is set to where funwave.exe resides
            cd(fullfile(string(app.directory)));



            %In reclicking the button, delete existing Makefile and funwave.exe
            sourceFileExe = fullfile(app.directory, "funwave");
            if exist(sourceFileExe,'file')
                delete(sourceFileExe);
            end


            % Determine the MPIF90 path based on user input or system defaults
            if isempty(app.Mpif90EditField.Value) || strcmp(app.Mpif90EditField.Value, '0') %|| ~exist(fullfile(app.Mpif90EditField.Value),"dir")
                % Set the default MPIF90 path based on the operating system
                if ismac
                    mpiPath = '/opt/homebrew/bin/';  % macOS default path
                elseif ispc
                    mpiPath = 'C:\Program Files (x86)\MPICH2\bin\';  % Windows default path
                else
                    error('Unsupported OS');  % Handle other operating systems or configurations
                end
                app.Mpif90EditField.Value = fullfile(mpiPath);  % Update the edit field with the default path
            else
                mpiPath = app.Mpif90EditField.Value;  % Use the user-provided MPIF90 path
            end


            if exist(fullfile(app.Mpif90EditField.Value),"dir") == 7 %Check if the mpif90 folder exists
                % Update the system PATH environment variable to include the MPI compiler path
                setenv('PATH', [getenv('PATH') ':' fileparts(mpiPath)]);
                app.Mpif90EditField.FontColor = 'k';

                if exist(fullfile(app.MakefileEditField.Value),"dir") == 7 %Check if the Makefile Source folder exists
                    cd(fullfile(app.MakefileEditField.Value));

                    % Check if the "Make Clean Completed" message is already in the log file
                    makeCleanMessage = '> Status: Make Clean...';
                    if ~contains(fileread(logFilePath), makeCleanMessage)
                        % Append the message only if it's not already there
                        fileId = fopen(logFilePath, 'a');
                        fprintf(fileId, '%s', makeCleanMessage);
                        drawnow
                    end

                    system('make clean');  % Clean up previous files created
                    completedMessage = '...Completed!';
                    if ~contains(logFilePath, completedMessage)
                        fileId = fopen(logFilePath, 'a');
                        fprintf(fileId, '%s\n\n', completedMessage);
                        fclose(fileId);
                        app.Preview_LogReport.Value = fileread(logFilePath); % Update the log report preview panel
                        drawnow;   % Update GUI
                    end

                else
                    fprintf(fileId, '%-30s%-s\n','MAKEFILE:', ' Invalid path. Missing .o, .F, and .f90 files');
                    fprintf(fileId, '%s\n', '');
                    app.Preview_LogReport.Value = fileread(logFilePath);
                    drawnow
                    %
                    % app.MakefileEditField.Value = "INVALID PATH";
                    % app.MakefileEditField.FontColor = 'w';
                    % app.MakefileEditField.BackgroundColor = 'r';
                end

                %--- Execute Makefile ---
                generatingFunwaveMessage = '> Status: Generating the funwave.exe... ';
                if ~contains(logFilePath, generatingFunwaveMessage)
                    fileId = fopen(logFilePath, 'a');
                    fprintf(fileId, '%s', generatingFunwaveMessage);
                    fclose(fileId);
                    app.Preview_LogReport.Value = fileread(logFilePath);
                    drawnow;
                end

                [status, cmdout] = system('make'); % Execute 'make' command

                if status ~= 0 %if make execution fails
                    if ~contains(logFilePath, 'Failed.')
                        fileId = fopen(logFilePath, 'a');
                        fprintf(fileId, '%s\n\n', 'Failed.');
                        app.Preview_LogReport.Value = fileread(logFilePath);
                        fclose(fileId);
                        drawnow
                    end

                    makeErrorMessaage = cmdout;
                    if ~contains(logFilePath, makeErrorMessaage)  %show the error message from the terminal
                        fileId = fopen(logFilePath, 'a');
                        fprintf(fileId, '%s\n', makeErrorMessaage);
                        fclose(fileId);
                        app.Preview_LogReport.Value = fileread(logFilePath);
                        drawnow;
                    end

                else %if successful, copy the funwave executable file to the specified output directory
                    sourceFile = fullfile(app.MakefileEditField.Value,
"funwave");
                    if exist(sourceFile,'file')
                        destinationFile = fullfile(app.directory, "funwave");
                        copyfile(sourceFile, destinationFile);
                        delete(sourceFile);  % Remove the source file after copying

                        %Append "Completed" message for "make" execution
                        if ~contains(logFilePath, 'Completed!')
                            fileId = fopen(logFilePath, 'a');
                            fprintf(fileId, '%s\n\n', 'Completed!');
                            app.Preview_LogReport.Value = fileread(logFilePath);
                            fclose(fileId);
                            drawnow
                        end
                        drawnow

                        MakefileMessage5 = '> Status: Starting the simulation...';
                        if ~contains(fileread(logFilePath), MakefileMessage5) %Successful status printed
                            % Append the message only if it's not already there
                            fileId = fopen(logFilePath, 'a');
                            fprintf(fileId, '%s\n\n', MakefileMessage5);
                            app.Preview_LogReport.Value =  fileread(logFilePath);

                            drawnow
                        end
                    else
                        if ~contains(logFilePath, 'Failed.')
                            fileId = fopen(logFilePath, 'a');
                            fprintf(fileId, '%s\n\n', 'Failed.');
                            app.Preview_LogReport.Value = fileread(logFilePath);
                            fclose(fileId);
                            drawnow
                        end
                    end
                end
            else
                % app.Mpif90EditField.Value = "INVALID PATH";
                % app.Mpif90EditField.FontColor = 'w';
                % app.Mpif90EditField.BackgroundColor = 'r';
                fileId = fopen(logFilePath, 'a');
                fprintf(fileId, [
                    '%-30s%-s\n' ...
                    '%-30s%s\n\n' ...
                    '%-30s%s\n' ...
                    '%-30s%s\n' ...
                    '%-40s%s\n\n' ...
                    '%-30s%s\n' ...
                    '%-40s%s\n\n' ...
                    '%-30s%s\n\n'
                    ], ...
                    'MPIF90:', 'Invalid MPI path - ''mpif90'' not found', ...
                    '', 'Failed to execute make', ...
                    '', 'In your computer''s terminal, type:', ...
                    '', '  *Mac OS: "which mpif90"', ...
                    '', '     Default: /opt/homebrew/bin/', ...
                    '', '  *Windows: "where mpif90"', ...
                    '', '     Default: C:\Program Files (x86)\MPICH2\bin', ...
                    '', 'Copy that path into the Mpif90 Folder textbox');

                app.Preview_LogReport.Value = fileread(logFilePath);
                fclose(fileId);
                drawnow
            end


            cd(fullfile(app.directory));


            % Append the start message only if it's not already in the log
            startMessage = sprintf('> Status: Starting the tsunami simulations...\n\n See the detailed progress in INPUT_FILES/LOG.txt');
            app.Preview_LogReport.Value = fileread(logFilePath);
            drawnow
            %
            if ~contains(logFilePath, startMessage)
                fileId = fopen(logFilePath, 'a');
                fprintf(fileId, '%s\n\n', startMessage);
                fclose(fileId); % Close the file after appending the start message
            end

            commandStr = ['mpirun -np ', num2str(app.ProcessorX.Value * app.ProcessorY.Value), ' ./funwave'];
            [status, cmdout] = system(commandStr);

            % Process cmdout to remove repeated lines and ensure unique logging
            cmdoutLines = splitlines(string(cmdout));
            [uniqueLines, ~] = unique(cmdoutLines, 'stable');
            uniqueCmdout = join(uniqueLines, newline);

            fileId = fopen(logFilePath, 'a'); % Reopen log file to append the execution status and cmdout
            if status == 0
                runCompletedMessage = 'FUNWAVE.EXE: Run Completed';
                if ~contains(logFilePath, runCompletedMessage)
                    fprintf(fileId, '%-30s%-s\n\n', runCompletedMessage);
                end
                for line = uniqueLines'
                    if line ~= ""
                        fprintf(fileId, '   %-28s\n', line);
                    end
                end
            else
                executionFailedMessage = 'FUNWAVE.EXE: Execution failed';
                if ~contains(currentLogContent, executionFailedMessage)
                    fprintf(fileId, '%-30s%-s\n', executionFailedMessage);
                end
```

```
                for line = uniqueLines'
                    if line ~= ""
                        fprintf(fileId, '   Error: %-25s\n', line);
                    end
                end
            end

            fclose(fileId); % Close the file


            app.Preview_LogReport.Value = fileread(logFilePath); % Display the updated log content



        end

        % Button pushed function: CreateMakefileButton
        function CreateMakefileButtonPushed(app, event)
            %Disable the simulation button
            app.StartsimulationButton.Enable = 'off';
            drawnow

            %Update the input.txt file
            GenerateinputtxtButtonPushed(app, event);

            %Switch tab of the Preview Panel to the Log Report Tab
            app.TabGroup3.SelectedTab = app.LogReportTab;

            %Locate the Log Report file
            cd(fullfile(app.directory));
            LogFolder = fullfile(app.directory, 'Log_Files');

            %If'INPUT_FILES' directory does not exist, create the folder
            if ~exist(LogFolder, 'dir')
                mkdir(LogFolder);
            end

            %If the log text file does not exist, create the text file
            logFilePath = fullfile(LogFolder,'Log_Report_InputPrep.txt');
            if ~exist(logFilePath,'file')
                fileId=fopen(logFilePath, 'w');
            else
                %Update the log report
                MakefileMessage00 = '******* MAKEFILE SECTION *******'; % Check if the "Creating Makefile completed" message is already in the log file
                if ~contains(fileread(logFilePath), MakefileMessage00)
                    % Append the message only if it's not already there
                    fileId = fopen(logFilePath, 'a');
                    fprintf(fileId, '%s\n\n', MakefileMessage00);
                    fprintf('');
                    app.Preview_LogReport.Value =  fileread(logFilePath);
                    drawnow
                end
            end



            %Remove pre-existing progress status under the Makefile section
            %Read the content of the log report
            fileContent = fileread(logFilePath);

            % Find the index of 'Makefile Section' header
            startIndex = strfind(fileContent, MakefileMessage00);

            %If there are pre-existing data in this section, overwrite them
            if ~isempty(startIndex)
                %Find the end of the line of the header
                endOfLineIndex = strfind(fileContent(startIndex:end),↙
    newline);
                endIndex = startIndex + endOfLineIndex(1) - 1;

                %Trim the fileContent up to endIndex
                trimmedContent = fileContent(1:endIndex);

                %Write the trimmed content back to the file
                fileId = fopen(logFilePath, 'w');
                fprintf(fileId, '%s', trimmedContent);
                fclose(fileId);

                %Update app.Preview_LogReport.Value with the updated content
                app.Preview_LogReport.Value = trimmedContent;
            else
                fprintf('MakefileMessage00 not found in %s\n', logFilePath);
            end

            %Update the log reporrt
            app.Preview_LogReport.Value = fileread(logFilePath);


            %↙
    ---------------------------------------------------------------------
            %------TEMPLATE OF THE MAKEFILE from FUNWAVE-TVD-version_3.↙
    6/src------
            %↙
    ---------------------------------------------------------------------
            makefile_template = {
                '#-----------BEGIN MAKEFILE---------------------------'
                'FUNWAVE_DIR = .'
                'WORK_DIR    = .'
                'COMPILER    = #COMPILER_OPTION#'
                'PARALLEL    = #PARALLEL_OPTION#'
                'EXEC        = funwave'
                'PRECISION   = #PRECISION_OPTION#'
                ''
                '#-----------DEFINE FLAGS------------------------------'
                '#         uncomment to choose the model'
                '#FLAG_1  = -DCOUPLING'
                '#FLAG_2  = -DZALPHA'
                '#FLAG_3  = -DVESSEL'
                '#FLAG_4  = -DMETEO'
                '#FLAG_5  = -DWIND'
                '#FLAG_6  = -DSEDIMENT'
                '#FLAG_7  = -DCHECK_MASS_CONSERVATION'
                '#FLAG_8  = -DTMP'
                '#FLAG_9 = -DTRACKING'
```

```
                     '#FLAG_10 = -DDEEP_DRAFT_VESSEL'
                     '#FLAG_11 = -DMIXING'
                     '#----------------uncommon options---------------------'
                     'DEF_FC      = mpif90'
                     'DEF_FC_FLAG = '
                     'SPHERICAL   = #COORDINATE_OPTION#'
                     'MPI         = openmpi'
                     'DEBUG       = true'
                     'INCS        = $(IOINCS) $(GOTMINCS)'
                     'LIBS        = $(PV3LIB) $(CLIB)  $(PARLIB) $(IOLIBS) ↙
$(MPILIB) $(GOTMLIB)'
                     'CLIB        ='
                     'MDEPFLAGS   = --cpp --fext=f90 --file=-'
                     'RANLIB      = ranlib'
                     ''
                     '#----------include the essential makefiles------------'
                     '#Copied file from $(FUNWAVE_DIR)/..↙
/GNUMake/Essential/Make_Essential'
                     '#--------------------Use shell------------------------'
                     'SHELL    = /bin/sh'
                     ''
                     '#-----Eliminate the leading and ending white space----'
                     'FUNWAVE_DIR := $(strip $(FUNWAVE_DIR))'
                     'WORK_DIR    := $(strip $(WORK_DIR))'
                     'COMPILER    := $(strip $(COMPILER))'
                     'PARALLEL    := $(strip $(PARALLEL))'
                     'PRECISION   := $(strip $(PRECISION))'
                     'EXEC        := $(strip $(EXEC))'

                     'DEF_FC      := $(strip $(DEF_FC))'
                     'DEF_FC_FLAG := $(strip $(DEF_FC_FLAG))'
                     'MPI         := $(strip $(MPI))'
                     'SPHERICAL   := $(strip $(SPHERICAL))'
                     'DEBUG       := $(strip $(DEBUG))'
                     ''
                     '#----------------C Pre-processor----------------------'
                     '#  note: CPP is not C++'
                     'CPP       = /usr/bin/cpp'
                     'CPPFLAGS = -P -traditional'
                     ''
                     '#----------------Fortran Compiler---------------------'
                     '# if $(DEF_FC) is not empty'
                     'ifneq ($(DEF_FC),$(filter $(DEF_FC), ''''))'
                     '   FC = $(DEF_FC)'
                     '# if $(DEF_FC) is empty'
                     'else'
                     '   # intel'
                     '   ifeq ($(COMPILER),$(filter $(COMPILER), intel))'
                     '      USE_MOD = -module $(MOD_DIR)'
                     '      ifeq ($(PARALLEL),$(filter $(PARALLEL), true))'
                     '         ifeq ($(MPI),$(filter $(MPI), intelmpi))'
                     '            FC = mpiifort'
                     '         else'
                     '            FC = mpif90'
                     '         endif'
                     '      else ifeq ($(PARALLEL),$(filter $(PARALLEL), false))'
                     '         FC = ifort'
                     '      endif'
                     '   # gnu'
                     '   else ifeq ($(COMPILER),$(filter $(COMPILER), gnu))'
                     '      USE_MOD = -J $(MOD_DIR)'
                     '      ifeq ($(PARALLEL),$(filter $(PARALLEL), true))'
                     '         FC = mpif90'
                     '      else ifeq ($(PARALLEL),$(filter $(PARALLEL), false))'
                     '         FC = gfortran'
                     '      endif'
                     '   # pgi'
                     '   else ifeq ($(COMPILER),$(filter $(COMPILER), pgi))'
                     '      USE_MOD = -module $(MOD_DIR)'
                     '      ifeq ($(PARALLEL),$(filter $(PARALLEL), true))'
                     '         FC = mpif90'
                     '      else ifeq ($(PARALLEL),$(filter $(PARALLEL), false))'
                     '         FC = pgfortran'
                     '      endif'
                     '   # onyx hpc'
                     '   else ifeq ($(COMPILER),$(filter $(COMPILER), onyx))'
                     '      USE_MOD = -module $(MOD_DIR)'
                     '      FC      = ftn'
                     '      # Removing -lmpi flag if defined in CLIB'
                     '      ifneq (, $(findstring -lmpi, $(CLIB)))'
                     '         $(warning WARNING: lmpi flag detected in CLIB, Onyx does not require explicit linking of MPI libary. Removing -lmpi from CLIB. )'
                     '         CLIB := $(filter-out -lmpi, $(CLIB))'
                     '      endif'
                     '      # Removing -Bdynamic flat if defined in CLIB'
                     '      ifneq (, $(findstring -Bdynamic, $(CLIB)))'
                     '         $(warning WARNING: Bdynamic flag detected in CLIB, Onyx executables do not work correctly with dynamic linking of shared libaries. Removing -Bdynamic from CLIB. )'
                     '         CLIB := $(filter-out -Bdynamic, $(CLIB))'
                     '      endif'


                     '   # koehr and gaffney'
                     '   else ifeq ($(COMPILER),$(filter $(COMPILER), koehr gaffney))'
                     '      USE_MOD = -module $(MOD_DIR)'
                     '      FC      = ifort'
                     '   # invalid option'
                     '   else'
                     '      $(error Fatal ERROR: COMPILER=$(COMPILER) and DEF_FC is empty, Please correct the COMPILER or customize the DEF_FC.)'
                     '   endif'
                     'endif'
                     ''
                     '#----------Debug and optimization flags---------------'
                     '# DEBUG=true'
                     'ifeq ($(DEBUG),$(filter $(DEBUG), true))'
                     '   ifeq ($(COMPILER),$(filter $(COMPILER), intel onyx gaffney koehr))'
                     '      DEBFLGS   = -g -check -warn -fPIC'
                     '      OPT       = -O0'
                     '   else ifeq ($(COMPILER),$(filter $(COMPILER), gnu))'
                     '      DEBFLGS   = -g -fcheck=all -Wall -fPIC'
                     '      OPT       = -O0'
                     '   else ifeq ($(COMPILER),$(filter $(COMPILER), pgi))'
                     '      DEBFLGS  = -g -fPIC'
                     '      OPT      = -O0'
                     '   endif'
                     'endif'
                     '# DEBUG=false'
                     'ifeq ($(DEBUG),$(filter $(DEBUG), false))'
                     '   ifeq ($(COMPILER),$(filter $(COMPILER), intel onyx gaffney koehr))'
                     '      DEBFLGS  = -fPIC'
                     '      OPT      = -O2'
                     '   else ifeq ($(COMPILER),$(filter $(COMPILER), gnu))'
                     '      DEBFLGS  = -fPIC'
                     '      OPT      = -O3'
                     '   else ifeq ($(COMPILER),$(filter $(COMPILER), pgi))'
                     '      DEBFLGS  = -w -fPIC'
                     '      OPT      = -O2'
                     '   endif'
                     'endif'
                     ''
                     '#---------------Pre-processor flags-------------------'
                     'ifeq ($(PRECISION),$(filter $(PRECISION), double))'
                     '   FLAG_PRECISION = -DDOUBLE_PRECISION '
                     'else ifeq ($(PRECISION),$(filter $(PRECISION), single))'
                     '   FLAG_PRECISION = '
                     'else'
                     '   $(error Fatal ERROR: PRECISION=$(PRECISION), should be single or double.)'
                     'endif'
                     ''
                     'ifeq ($(SPHERICAL),$(filter $(SPHERICAL), true))'
                     '   FLAG_CARTESIAN = '
                     'else'
                     '   FLAG_CARTESIAN = -DCARTESIAN'
                     'endif'
                     ''
                     'ifeq ($(PARALLEL),$(filter $(PARALLEL), true))'
                     '   FLAG_PARALLEL = -DPARALLEL'
                     'endif'
                     ''
                     'ifeq ($(COMPILER),$(filter $(COMPILER), intel onyx koehr gaffney))'
                     '   FLAG_INTEL = -DINTEL'
                     'endif'
                     ''
                     'ifeq ($(DEBUG),$(filter $(DEBUG), true))'
                     '   FLAG_DEBUG = -DDEBUG'
                     'endif'
                     '#----------------FUNWAVE flags-----------------------'
                     'FLAG_ARGS = $(FLAG_1) $(FLAG_2) \'
                     '           $(FLAG_3) $(FLAG_4) $(FLAG_5) $(FLAG_6) \'
                     '           $(FLAG_7) $(FLAG_8) $(FLAG_9) $(FLAG_10)  \'
                     '           $(FLAG_11) $(FLAG_12) $(FLAG_13) $(FLAG_14) \'
                     '           $(FLAG_15) $(FLAG_16) $(FLAG_17) $(FLAG_18) \'
                     '           $(FLAG_19) $(FLAG_20) $(FLAG_21) $(FLAG_22) \'
                     '           $(FLAG_23) $(FLAG_24)'
                     'CPPARGS = $(CPPFLAGS) $(DEF_FLAGS) $(FLAG_ARGS) \'
                     '           $(FLAG_PARALLEL) $(FLAG_INTEL) $(FLAG_DEBUG) \'
                     '           $(FLAG_PRECISION) $(FLAG_CARTESIAN)'
```

```
''
'# remove extra whitespace between flags'
'CPPARGS := $(shell echo $(CPPARGS) | sed ''s~ \+~ ~g'')'
''
'#---------------------EXEC name------------------------'
'EXEC := $(WORK_DIR)/$(EXEC)'
''
'#-------------------Library name-----------------------'
'ifneq ($(DEF_FC),$(filter $(DEF_FC), ''))'
'   FCFLAGS := $(DEF_FC_FLAG) $(INCS)'
'else'
'   FCFLAGS := $(DEBFLGS) $(OPT) $(INCS)'
'endif'
''
'# remove extra whitespace between flags'
'FCFLAGS := $(shell echo $(FCFLAGS) | sed ''s~ \+~ ~g'')'
''
'#----------------FUNWAVE souce code-------------------'
'MODS  = mod_param.F mod_global.F mod_input.F mod_vessel.F mod_bathy_correction.F \'
'        mod_meteo.F mod_parallel_field_io.F mod_sediment.F mod_tide.F\'
'        mod_tracer.F'
''
'MAIN  = main.F bc.F fluxes.F init.F io.F tridiagnal.F \'
'        breaker.F derivatives.F dispersion.F etauv_solver.F \'
'        sponge.F sources.F masks.F parallel.F statistics.F \'
'        wavemaker.F mixing.F nesting.F misc.F samples.F\'
''
'#-----------------Compile process---------------------'
'SRC_DIR   = $(FUNWAVE_DIR)'
'PRE_DIR   = $(WORK_DIR)/build/pre'
'OBJ_DIR   = $(WORK_DIR)/build/obj'
'MOD_DIR   = $(WORK_DIR)/build/mod'
''
'SRCS := $(MODS)  $(MAIN)'
'SRCS := $(patsubst %,$(SRC_DIR)/%,$(SRCS))'
'PRES := $(patsubst $(SRC_DIR)/%.F,$(PRE_DIR)/%.f90,$(SRCS))'
'OBJS := $(patsubst $(PRE_DIR)/%.f90,$(OBJ_DIR)/%.o,$(PRES))'
''
'# keep the *.f90 files in $(PRE_DIR) after make'
'# .PRECIOUS: $(PRE_DIR)/%.f90'
''
'# pre-processors'
'$(PRE_DIR)/%.f90 : $(SRC_DIR)/%.F'
'   $(CPP) $(CPPARGS) $< $@'
''
'# compile objective files'
'$(OBJ_DIR)/%.o : $(PRE_DIR)/%.f90'
'   $(FC) $(FCFLAGS) -c $< -o $@ $(USE_MOD)'
''
'# link and generate the exec file'
'$(EXEC): $(OBJS)'
'   $(FC) $(FCFLAGS) -o $(EXEC) $(OBJS) $(LIBS)'
''
'$(PRES): | $(PRE_DIR)'
```

```
            ''
            '$(PRE_DIR):'
            '   mkdir -p $(WORK_DIR)'
            '   mkdir -p $(PRE_DIR)'
            '   mkdir -p $(OBJ_DIR)'
            '   mkdir -p $(MOD_DIR)'
            '#---------------Cleaning targets-----------------------'
            'clean:'
            '       /bin/rm -rf $(WORK_DIR)/build'
            ''
            'clobber: clean'
            '       /bin/rm -f $(EXEC)'
            ''
            'extra-clobber: clean'
            '       /bin/rm -rf $(WORK_DIR)'
            ''};


        mkfile = sprintf('%s\n',makefile_template{:});
        mkfile = string(mkfile);


        %--------------------------------------------------
        %------Uncommenting the Flags in the Makefile------
        %--------------------------------------------------

        %------DOUBLE PRECISION------
        if app.DoublePrecisionCheckBox.Value
            mkfile = strrep(mkfile, "#PRECISION_OPTION#", "double");
        else
            mkfile = strrep(mkfile, "#PRECISION_OPTION#", "single");
        end

        %------PARALLEL MODE------
        if app.ParallelModeCheckBox.Value
            mkfile = strrep(mkfile, "#PARALLEL_OPTION#","true");
        else
            mkfile = strrep(mkfile, "#PARALLEL_OPTION#","false");

        end

        %------INTEL COMPILER------
        if app.IntelCompilerCheckBox.Value
            mkfile  = strrep(mkfile, "#COMPILER_OPTION#", "intel");
        else
            mkfile  = strrep(mkfile, "#COMPILER_OPTION#", "gnu");
        end


        %------GRID------
        if app.SphericalButton.Value
            %     mkfile = strrep(mkfile,"#FLAG_2","FLAG_2");  %Ask Masashi if it is always true -DZALPHA
            mkfile = strrep(mkfile,"#COORDINATE_OPTION#", "true");
        elseif app.CartesianButton.Value
            mkfile = strrep(mkfile,"#COORDINATE_OPTION#", "false");
        end

```

```

            %------TURBULENT MIXING------
            if app.UseSmagorinskyCheckBox.Value
                mkfile = strrep(mkfile,"#FLAG_12","FLAG_12");
            end


            %------METEOTSUNAMI------
            if app.MeteotsunamiButton.Value
                mkfile = strrep(mkfile,"#FLAG_5","FLAG_5");
                % Checking if any of the switches are "On" to perform the operation once
                if strcmp(app.WindforceSwitch.Value, "On") || strcmp(app.HollandstormmodelSwitch.Value, "On") || strcmp(app.ConstantwindfieldSwitch.Value, "On")
                    mkfile = strrep(mkfile, "#FLAG_6", "FLAG_6");
                end
            end

            %------SEDIMENT------
            if app.SedimentTransportCheckBox.Value
                mkfile = strrep(mkfile,"#FLAG_7","FLAG_7");
            end

            %------VESSEL------
            if app.VesselButton.Value
                mkfile = strrep(mkfile, "#FLAG_4", "FLAG_4");

                if app.IncludeSedimentEffectintheModelCheckBox.Value
                    mkfile = strrep(mkfile,"#FLAG_7","FLAG_7");
                end

                % Deep-draft Vessel
                if app.ActivateCheckBox.Value
                    mkfile = strrep(mkfile, "#FLAG_11", "FLAG_11");
                end
            end


            %------------------------------------------------------------
            %------Create the Makefile using the Makefile template------
            %------------------------------------------------------------
            outputFilePath3 = fullfile(app.directory, 'Makefile');
            outputFile3 = fopen(outputFilePath3, 'w');

            for i = 1:length(mkfile)
                fprintf(outputFile3, '%s\n',mkfile{i});
            end
            fclose(outputFile3);
            app.Preview_Makefile.Value = string(mkfile);

            %Update the log report -  Check if the "Creating Makefile completed" message is already in the log file
            MakefileMessage = '> Status: Creating the Makefile in the INPUT_FILES folder... Completed!';
            if ~contains(fileread(logFilePath), MakefileMessage)
                % Append the message only if it's not already there
                fileId = fopen(logFilePath, 'a');
                fprintf(fileId, '\n %s\n\n', MakefileMessage);
                app.Preview_LogReport.Value =  fileread(logFilePath);
                drawnow
            end

            %Texts to include in the log report
            MakefileMessage2 = '> Status: Copying the Makefile into the Makefile source folder... Completed!';
            MakefileMessage3 = '> Status: Copying the Makefile into the Makefile source folder... Failed.';
            MakefileMessage4 = '> Invalid path for the Makefile source';

            %---------------------------------------------------------------
            %------Copy the generated Makefile into the source folder------
            %---------------------------------------------------------------
            %If the Makefile source folder exists, copy the in-app generated Makefile to the Makefile source folder
            if exist(fullfile(app.MakefileEditField.Value),"dir") == 7
                cd(fullfile(app.MakefileEditField.Value));
                sourceFile2 = fullfile(app.directory, "Makefile");
                destinationFile2 = fullfile(app.MakefileEditField.Value);
                copyfile(sourceFile2, destinationFile2);

                %Enable the MPIf90 folder & the 'Start Simulation' button
                app.Mpif90EditField.Enable = 'on';
                app.Mpif90EditFieldLabel.Enable = 'on';
                app.Button_41.Enable = 'on';
                app.StartsimulationButton.Enable = "on";
                drawnow

                %Update the log report to show that the copying of file is completed
                if ~contains(fileread(logFilePath), MakefileMessage2)
                    fileId = fopen(logFilePath, 'a');
                    fprintf(fileId, '%s\n\n', MakefileMessage2);
                    app.Preview_LogReport.Value =  fileread(logFilePath);
                    drawnow
                end

            else %Print error status in the report
                if ~contains(fileread(logFilePath), MakefileMessage3)
                    fileId = fopen(logFilePath, 'a');
                    fprintf(fileId, '%s\n\n', MakefileMessage3);
                    fprintf(fileId, '%s\n\n', MakefileMessage4);
                    app.Preview_LogReport.Value =  fileread(logFilePath);
                    drawnow
                end

                app.Preview_LogReport.Value = fileread(logFilePath);
                app.StartsimulationButton.Enable = "off";

                app.MakefileEditField.Value = "INVALID PATH";
                app.MakefileEditField.FontColor = 'w';
                app.MakefileEditField.BackgroundColor = 'r';
                drawnow
            end

            %Enable the Makefile and input.txt buttons
```

```
            app.CreateMakefileButton.Enable = 'on';
            app.GenerateinputtxtButton.Enable = 'on';
            drawnow
        end

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

        end

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

        end

        % Value changed function: FrictionCheckBox_2
        function FrictionCheckBox_2ValueChanged(app, event)
            if app.FrictionCheckBox_2.Value
                app.FrictionCoefficientEditField.Enable = 'on';
                app.FrictionCoefficientEditFieldLabel.Enable = 'on';
            else
                app.FrictionCoefficientEditField.Enable = 'off';
                app.FrictionCoefficientEditFieldLabel.Enable = 'off';
            end
        end

        % Value changed function: ViscosityCheckBox
        function ViscosityCheckBoxValueChanged(app, event)
            if app.ViscosityCheckBox.Value
                app.ValueEditField.Enable = 'on';
                app.ValueEditFieldLabel_2.Enable = 'on';
            else
                app.ValueEditField.Enable = 'off';
                app.ValueEditFieldLabel_2.Enable = 'off';
            end

        end

        % Value changed function: ActivateCheckBox
        function ActivateCheckBoxValueChanged(app, event)
            if app.ActivateCheckBox.Value
                if app.FrictionCheckBox_2.Value
                    app.FrictionCoefficientEditField.Enable = 'on';
                    app.FrictionCoefficientEditFieldLabel.Enable = 'on';
                else
                    app.FrictionCoefficientEditField.Enable = 'off';
                    app.FrictionCoefficientEditFieldLabel.Enable = 'off';
                end

                if app.ViscosityCheckBox.Value
                    app.ValueEditField.Enable = 'on';
                    app.ValueEditFieldLabel_2.Enable = 'on';
                else
                    app.ValueEditField.Enable = 'off';
                    app.ValueEditFieldLabel_2.Enable = 'off';
                end
```

```

               app.SelectMethodLabel.Enable = 'on';
               app.MinimumClearanceEditField.Enable = 'on';
               app.MinimumClearanceLabel.Enable = 'on';
               app.ViscosityCheckBox.Enable = 'on';
               app.FrictionCheckBox_2.Enable = 'on';
               app.ShockcapturingCheckBox.Enable = 'on';
            else
                app.SelectMethodLabel.Enable = 'off';
                app.MinimumClearanceEditField.Enable = 'off';
                app.MinimumClearanceLabel.Enable = 'off';
                app.ValueEditField.Enable = 'off';
                app.ValueEditFieldLabel_2.Enable = 'off';
                app.FrictionCoefficientEditField.Enable = 'off';
                app.FrictionCoefficientEditFieldLabel.Enable = 'off';
                app.ViscosityCheckBox.Enable = 'off';
                app.FrictionCheckBox_2.Enable = 'off';
                app.ShockcapturingCheckBox.Enable = 'off';
            end
        end

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

        end

        % Button pushed function: GenerateinputtxtButton_2
        function GenerateinputtxtButton_2Pushed(app, event)
%            app.GenerateinputtxtButtonPushed(app, []);
           GenerateinputtxtButtonPushed(app, event);
        end

        % Button pushed function: Button_42
        function Button_42Pushed2(app, event)
            workingfolder = uigetdir;
            workingfolder = fullfile(workingfolder,' ');
            app.MakefileEditField.Value = deblank(string(workingfolder));
            app.MakefileEditField.FontColor = 'k';
            app.MakefileEditField.BackgroundColor = 'w';
        end

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

        end

        % Callback function
        function GenerateinputtxtButton_3Pushed(app, event)
                 GenerateinputtxtButtonPushed(app, event);
        end

        % Selection changed function: ButtonGroup_23
        function ButtonGroup_23SelectionChanged(app, event)

            if app.ShockwavecapturingButton.Value
                app.C1EditField.Value = 0.65;
                app.C2EditField.Visible= 'off';
                app.C2EditFieldLabel.Visible = 'off';
                app.C1EditField.Visible = 'off';
                app.C1EditFieldLabel.Visible = 'off';
            elseif app.ViscositybreakingButton.Value
                app.C2EditField.Visible = "on";
                  app.C2EditField.Enable = "on";
                app.C2EditFieldLabel.Visible = "on";
                 app.C2EditFieldLabel.Enable = "on";
                app.C1EditField.Visible = "on";
                app.C1EditField.Enable = "on";
                app.C1EditFieldLabel.Visible = "on";
                app.C1EditFieldLabel.Enable = "on";

            end
        end

        % Value changed function: InvertvaluesCheckBox
        function InvertvaluesCheckBoxValueChanged(app, event)


        end

        % Callback function
        function ImportInfoButtonPushed(app, event)
            [bathymetryfile,path2] = uigetfile('*.tiff;*.tif');
            bathymetryinputdata = string(path2) + string(bathymetryfile);


            if contains(bathymetryinputdata,'.tif')
                try
                    app.depthtype = 2;
                    app.LatitudeEditField.Enable = "on";
                    app.LongitudeEditField.Enable = "on";
                    %                 try

                    [A1,R2] = readgeoraster(bathymetryinputdata);
                    A =  size(A1);

                    %Extract M and N global dimenstions
                    app.XEditField_2.Value = A(1,2);
                    app.YEditField_2.Value = A(1,1);
                    app.XEditField_2.FontColor = 'k';
                    app.YEditField_2.FontColor = 'k';

                    %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 of the bathymetry
                    app.XEditField.Value = R2.CellExtentInLatitude;
                    app.YEditField.Value = R2.CellExtentInLongitude;
                    app.XEditField.FontColor = 'k';
                    app.YEditField.FontColor = 'k';

                    if R2.AngleUnits == "degrees"
                        app.ButtonGroup_4.SelectedObject = app.↙
SphericalButton;
                        app.SphericalButton.FontColor = 'k';
                        app.CartesianButton.FontColor = 'k';
                    end
                    app.rasterdegrees = R2;

                catch
                    app.ButtonGroup_4.SelectedObject = app.Button_2;
                end
            end
        end

        % Value changed function: DoublePrecisionCheckBox
        function DoublePrecisionCheckBoxValueChanged(app, event)


        end

        % Button pushed function: BoundarylimitsButton
        function BoundarylimitsButtonPushed(app, event)
            [bathymetryfile,path2] = uigetfile(['*.tif;*.asc']);

            bathymetryinputdata = string(path2) + string(bathymetryfile);


            if contains(bathymetryinputdata,'.tif')
                %                  try
                app.depthtype = 2;

                [A1,R2] = readgeoraster(bathymetryinputdata);
                A =  size(A1);

                %Extract M and N global dimenstions
                app.CouplingX = R2.RasterSize(1,2);
                app.CouplingY = R2.RasterSize(1,1);

                %Extract grid size of the bathymetry
                cellsize = R2.CellExtentInLatitude;
                app.SpacingEditField.Value = cellsize;
                app.SpacingEditField.FontColor = 'k';

                % Extract and round the southwest corner coordinates
                rawSouth =min(R2.LatitudeLimits);
                westValue = min(R2.LongitudeLimits);



                % Process based on input type
                if isnumeric(rawSouth) && isscalar(rawSouth)
                    % Case 1: Scalar numeric input
                    southValue = rawSouth;

                elseif ischar(rawSouth) || isstring(rawSouth)
                    % Case 2: String input
                    southValue = str2double(rawSouth);

                elseif isnumeric(rawSouth) && ~isscalar(rawSouth)
                    % Case 3: Numeric array input
                    southValue = double(rawSouth);
                end

                % Assign to SouthEditField_2 if valid and within limits
                if exist('southValue', 'var') && isnumeric(southValue) && isscalar(southValue)
                    southLimits = app.SouthEditField_2.Limits;
                    if southValue >= southLimits(1) && southValue <= southLimits(2)
                        app.SouthEditField_2.Value = southValue;
                    end
                end

% Validate and assign west value
if isnumeric(westValue) && isscalar(westValue)
    westLimits = app.WestEditField_2.Limits;
    if westValue >= westLimits(1) && westValue <= westLimits(2)
        app.WestEditField_2.Value = westValue;
    end
end

app.EastEditField_2.Value = westValue + (cellsize*app.CouplingX);
app.NorthEditField_2.Value = southValue  + (cellsize*app.CouplingY);

            end
            %     % Initialize Xmatrix and Ymatrix
            %     Xmatrix = 1:app.CouplingX;
            %     Ymatrix = 1:app.CouplingY;
            %
            %     % Set the first element to the southwest corner
            %     Xmatrix(Xmatrix == 1) = min(R2.LongitudeLimits);
            %     Ymatrix(Ymatrix == 1) = min(R2.LatitudeLimits);
            %
            %     % Calculate CoupMatrixX and CoupMatrixY
            %     CoupMatrixX = Xmatrix * cellsize + min(R2.LongitudeLimits);
            %     CoupMatrixY = Ymatrix * cellsize + min(R2.LatitudeLimits);
            %
            %     % Recalculate Xmatrix and CoupMatrixX
            %     Xmatrix = 1:app.CouplingX;
            %     Xmatrix(Xmatrix == 1) = 0;
            %     for i = 1:length(Xmatrix)
            %         if Xmatrix(i) ~= 0
            %             Xmatrix(i) = Xmatrix(i) * cellsize;
            %         end
            %         CoupMatrixX(i) = Xmatrix(i) + min(R2.LongitudeLimits);
            %     end
            %
            %     % Recalculate Ymatrix and CoupMatrixY
            %     Ymatrix = 1:app.CouplingY;
            %     Ymatrix(Ymatrix == 1) = 0;
            %     for i = 1:length(Ymatrix)
            %         if Ymatrix(i) > 1
            %             Ymatrix(i) = Ymatrix(i) * cellsize;
            %         end
            %         CoupMatrixY(i) = Ymatrix(i) + min(R2.LatitudeLimits);
            %     end
            %
            %     % Store matrices
            %     app.CoupX = CoupMatrixX;
            %     app.CoupY = CoupMatrixY;
            %     app.CoupMatrixON = 0; % Flag for nested grid boundary setup
            %
            %     % Validate and assign east value
            %     eastValue = max(CoupMatrixX(:));
            %     if isnumeric(eastValue) && isscalar(eastValue)
            %         eastLimits = app.EastEditField_2.Limits;
            %         if eastValue >= eastLimits(1) && eastValue <= eastLimits(2)
            %             app.EastEditField_2.Value = eastValue;
            %         end
            %     end
            %
            %     % Validate and assign north value
            %     northValue = max(CoupMatrixY(:));
            %     if isnumeric(northValue) && isscalar(northValue)
            %         northLimits = app.NorthEditField_2.Limits;
            %         if northValue >= northLimits(1) && northValue <= northLimits(2)
            %             app.NorthEditField_2.Value = northValue;
            %         end
            %     end
            %
                app.SouthEditField_2.FontColor = 'k';
                app.WestEditField_2.FontColor = 'k';
                app.NorthEditField_2.FontColor = 'k';
                app.EastEditField_2.FontColor = 'k';
                drawnow
            % end
            %




        end

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

        end

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

        end

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

        % Value changed function: WestEditField_2
        function WestEditField_2ValueChanged(app, event)
```

```
            app.WestEditField_2.FontColor = 'k';

        end

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

        end

        % Size changed function: Panel_32
        function Panel_32SizeChanged(app, event)


        end

        % Button pushed function: GenerateinputtxtButton_4
        function GenerateinputtxtButton_4Pushed(app, event)
              GenerateinputtxtButtonPushed(app, event);
        end

        % Button pushed function: Button_43
        function Button_43Pushed(app, event)
                [file,path2] = uigetfile('*.*');
                app.PressurefileEditField.Value = string(file);
                app.PressurefileEditField.FontColor = 'k';
                app.PressurefileEditField.BackgroundColor = 'w';
                app.PressureFileLOC = string(path2) + string(file);
                app.PressureFileNAME = string(file);
        end

        % Value changed function: SedimentTransportCheckBox
        function SedimentTransportCheckBoxValueChanged(app, event)
            %Set some 'Expected Output' parameters to T or F
            if app.SedimentTransportCheckBox.Value
                app.TabGroup2.SelectedTab = app.Tab_3;
                app.DepthCheckBox.Value = 1;
                app.UVelocityCheckBox.Value = 1;
                app.VVelocityCheckBox.Value = 1;
                app.WetdrymaskCheckBox.Value = 1;
                app.WaveHeightCheckBox.Value = 1;
            else
                app.DepthCheckBox.Value = 0;
                app.UVelocityCheckBox.Value = 0;
                app.VVelocityCheckBox.Value = 0;
                app.WetdrymaskCheckBox.Value = 0;
                app.WaveHeightCheckBox.Value = 0;
            end


        end

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

        end

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

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

        end

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

        end

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

        end

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

        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 = [100 100 1146 676];
            app.UIFigure.Name = 'MATLAB App';
            app.UIFigure.Resize = 'off';
            app.UIFigure.Scrollable = 'on';

            % Create TabGroup3
            app.TabGroup3 = uitabgroup(app.UIFigure);
            app.TabGroup3.AutoResizeChildren = 'off';
            app.TabGroup3.Position = [743 128 399 547];

            % Create LogReportTab
            app.LogReportTab = uitab(app.TabGroup3);
            app.LogReportTab.AutoResizeChildren = 'off';
            app.LogReportTab.Title = 'Log Report';

            % Create Preview_LogReport
            app.Preview_LogReport = uitextarea(app.LogReportTab);
            app.Preview_LogReport.HandleVisibility = 'off';
            app.Preview_LogReport.Editable = 'off';
            app.Preview_LogReport.WordWrap = 'off';
            app.Preview_LogReport.Position = [1 5 397 506];

            % Create inputtxtTab
            app.inputtxtTab = uitab(app.TabGroup3);
            app.inputtxtTab.AutoResizeChildren = 'off';
            app.inputtxtTab.Title = 'input.txt';

            % Create Preview_Input
            app.Preview_Input = uitextarea(app.inputtxtTab);
            app.Preview_Input.Editable = 'off';
            app.Preview_Input.Position = [1 0 397 511];

            % Create Makefile
            app.Makefile = uitab(app.TabGroup3);
            app.Makefile.AutoResizeChildren = 'off';
            app.Makefile.Title = 'Makefile';

            % Create Preview_Makefile
            app.Preview_Makefile = uitextarea(app.Makefile);
            app.Preview_Makefile.Editable = 'off';
            app.Preview_Makefile.Position = [1 5 397 506];

            % Create TabGroup
            app.TabGroup = uitabgroup(app.UIFigure);
            app.TabGroup.AutoResizeChildren = 'off';
            app.TabGroup.Position = [1 1 737 674];

            % Create InputTab
            app.InputTab = uitab(app.TabGroup);
            app.InputTab.AutoResizeChildren = 'off';
            app.InputTab.SizeChangedFcn = createCallbackFcn(app, @InputTabSizeChanged, true);
            app.InputTab.Title = 'Input';
            app.InputTab.BackgroundColor = [0.9412 0.9412 0.9412];
            app.InputTab.ForegroundColor = [0.149 0.149 0.149];
            app.InputTab.Scrollable = 'on';

            % Create Panel_27
            app.Panel_27 = uipanel(app.InputTab);
            app.Panel_27.AutoResizeChildren = 'off';
            app.Panel_27.Position = [371 396 356 42];

            % Create WetDrySchemeEditFieldLabel
            app.WetDrySchemeEditFieldLabel = uilabel(app.Panel_27);
            app.WetDrySchemeEditFieldLabel.Tooltip = {'Minimum Depth (m) for wet-dry scheme.'; ''; 'Suggested values:'; 'Lab scale: 0.001'; 'Field scale: 0.01'};
            app.WetDrySchemeEditFieldLabel.Position = [25 7 97 22];
            app.WetDrySchemeEditFieldLabel.Text = 'Wet-Dry Scheme';

            % Create WetDrySchemeEditField
            app.WetDrySchemeEditField = uieditfield(app.Panel_27, 'numeric');
            app.WetDrySchemeEditField.Limits = [0 Inf];
            app.WetDrySchemeEditField.ValueDisplayFormat = '%.2f';
            app.WetDrySchemeEditField.ValueChangedFcn = createCallbackFcn(app, @WetDrySchemeEditFieldValueChanged, true);
            app.WetDrySchemeEditField.FontColor = [0.651 0.651 0.651];
            app.WetDrySchemeEditField.Tooltip = {''};
            app.WetDrySchemeEditField.Position = [124 10 42 16];
            app.WetDrySchemeEditField.Value = 10;

            % Create BottomFrictionEditFieldLabel
            app.BottomFrictionEditFieldLabel = uilabel(app.Panel_27);
            app.BottomFrictionEditFieldLabel.HorizontalAlignment = 'right';
            app.BottomFrictionEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.BottomFrictionEditFieldLabel.Tooltip = {'Minimum Depth (m) to limit bottom friction.'; ''; 'Suggested values:'; 'Lab scale: 0.01'; 'Field scale: 0.1'};
            app.BottomFrictionEditFieldLabel.Position = [188 7 89 22];
            app.BottomFrictionEditFieldLabel.Text = 'Bottom Friction';

            % Create BottomFrictionEditField
            app.BottomFrictionEditField = uieditfield(app.Panel_27, 'numeric');
            app.BottomFrictionEditField.Limits = [0 Inf];
            app.BottomFrictionEditField.ValueDisplayFormat = '%11.2f';
            app.BottomFrictionEditField.ValueChangedFcn = createCallbackFcn(app, @BottomFrictionEditFieldValueChanged, true);
            app.BottomFrictionEditField.FontColor = [0.651 0.651 0.651];
            app.BottomFrictionEditField.Tooltip = {''};
            app.BottomFrictionEditField.Position = [284 10 41 16];
            app.BottomFrictionEditField.Value = 0.1;

            % Create Panel_25
            app.Panel_25 = uipanel(app.InputTab);
            app.Panel_25.AutoResizeChildren = 'off';
            app.Panel_25.Position = [371 522 356 116];

            % Create ButtonGroup_3
            app.ButtonGroup_3 = uibuttongroup(app.Panel_25);
            app.ButtonGroup_3.AutoResizeChildren = 'off';
            app.ButtonGroup_3.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_3SelectionChanged, true);
            app.ButtonGroup_3.BorderType = 'none';
            app.ButtonGroup_3.Position = [7 10 344 95];

            % Create LinearShallowWaterEquationButton
            app.LinearShallowWaterEquationButton = uiradiobutton(app.ButtonGroup_3);
            app.LinearShallowWaterEquationButton.Tooltip = {'Use linear shallow water equation.'; ''; 'When Boussinesq equation is not used, the ''HIGH_ORDER'' value under ''Numerics'' in the input.txt is set from ''THIRD'' to ''SECOND'' '};
            app.LinearShallowWaterEquationButton.Text = ' Linear Shallow Water Equation';
            app.LinearShallowWaterEquationButton.Position = [27 72 191 22];

            % Create FullyNonlinearBoussinesqEquationButton
            app.FullyNonlinearBoussinesqEquationButton = uiradiobutton(app.ButtonGroup_3);
            app.FullyNonlinearBoussinesqEquationButton.Text = ' Fully Nonlinear Boussinesq Equation';
            app.FullyNonlinearBoussinesqEquationButton.Position = [27 24 224 22];

            % Create Button
            app.Button = uiradiobutton(app.ButtonGroup_3);
            app.Button.Visible = 'off';
            app.Button.Text = '';
            app.Button.Position = [160 73 25 22];
            app.Button.Value = true;

            % Create NonlinearShallowWaterEquationButton
            app.NonlinearShallowWaterEquationButton = uiradiobutton(app.ButtonGroup_3);
            app.NonlinearShallowWaterEquationButton.Tooltip = {'Use nonlinear shallow water equation'; ''; ''; 'When Boussinesq equation is not used, the ''HIGH_ORDER'' value under ''Numerics'' in the input.txt is set from ''THIRD'' to ''SECOND'' '};
            app.NonlinearShallowWaterEquationButton.Text = ' Nonlinear Shallow Water Equation';
            app.NonlinearShallowWaterEquationButton.Position = [27 48 209 22];

            % Create FroudeCapEditFieldLabel
            app.FroudeCapEditFieldLabel = uilabel(app.ButtonGroup_3);
            app.FroudeCapEditFieldLabel.HorizontalAlignment = 'right';
            app.FroudeCapEditFieldLabel.Tooltip = {'Froude number limit for velocity efficiency'};
            app.FroudeCapEditFieldLabel.Position = [41 0 69 22];
            app.FroudeCapEditFieldLabel.Text = 'Froude Cap';

            % Create FroudeCapEditField
            app.FroudeCapEditField = uieditfield(app.ButtonGroup_3, 'numeric');
            app.FroudeCapEditField.Limits = [0 Inf];
            app.FroudeCapEditField.ValueDisplayFormat = '%8.2f';
            app.FroudeCapEditField.ValueChangedFcn = createCallbackFcn(app, @FroudeCapEditFieldValueChanged, true);
            app.FroudeCapEditField.FontColor = [0.651 0.651 0.651];
            app.FroudeCapEditField.Position = [119 3 41 16];
            app.FroudeCapEditField.Value = 10;

            % Create Panel_17
            app.Panel_17 = uipanel(app.InputTab);
            app.Panel_17.AutoResizeChildren = 'off';
            app.Panel_17.Position = [10 596 351 42];

            % Create Panel_5
            app.Panel_5 = uipanel(app.Panel_17);
            app.Panel_5.AutoResizeChildren = 'off';
            app.Panel_5.BorderType = 'none';
            app.Panel_5.Position = [43 4 264 30];

            % Create ProcessornumbersXLabel
            app.ProcessornumbersXLabel = uilabel(app.Panel_5);
            app.ProcessornumbersXLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.ProcessornumbersXLabel.HorizontalAlignment = 'right';
            app.ProcessornumbersXLabel.Tooltip = {'Must be consistent with mpirun -np n (px*py)'};
            app.ProcessornumbersXLabel.Position = [3 3 134 22];
            app.ProcessornumbersXLabel.Text = 'Processor numbers:    X';

```

```
        % Create ProcessorX
        app.ProcessorX = uieditfield(app.Panel_5, 'numeric');
        app.ProcessorX.Limits = [0 Inf];
        app.ProcessorX.ValueChangedFcn = createCallbackFcn(app, @ProcessorXValueChanged, true);
        app.ProcessorX.FontColor = [0.651 0.651 0.651];
        app.ProcessorX.Tooltip = {''};
        app.ProcessorX.Position = [146 6 34 16];
        app.ProcessorX.Value = 2;

        % Create YLabel
        app.YLabel = uilabel(app.Panel_5);
        app.YLabel.HorizontalAlignment = 'right';
        app.YLabel.Tooltip = {'Must be consistent with mpirun -np n(px*py)'};
        app.YLabel.Position = [180 3 35 22];
        app.YLabel.Text = 'Y';

        % Create ProcessorY
        app.ProcessorY = uieditfield(app.Panel_5, 'numeric');
        app.ProcessorY.ValueChangedFcn = createCallbackFcn(app, @ProcessorYValueChanged, true);
        app.ProcessorY.FontColor = [0.651 0.651 0.651];
        app.ProcessorY.Tooltip = {''};
        app.ProcessorY.Position = [223 6 32 16];
        app.ProcessorY.Value = 2;

        % Create ParallelInfoLabel
        app.ParallelInfoLabel = uilabel(app.InputTab);
        app.ParallelInfoLabel.BackgroundColor = [0.9412 0.9412 0.9412];
        app.ParallelInfoLabel.FontSize = 13;
        app.ParallelInfoLabel.FontWeight = 'bold';
        app.ParallelInfoLabel.FontColor = [0.0314 0.3686 0.6];
        app.ParallelInfoLabel.Position = [21 626 89 22];
        app.ParallelInfoLabel.Text = '  Parallel Info  ';

        % Create Friction________________________________Label_2
        app.Friction________________________________Label_2 = uilabel(app.InputTab);
        app.Friction________________________________Label_2.BackgroundColor = [0.9412 0.9412 0.9412];
        app.Friction________________________________Label_2.FontSize = 13;
        app.Friction________________________________Label_2.FontWeight = 'bold';
        app.Friction________________________________Label_2.FontColor = [0.0314 0.3686 0.6];
        app.Friction________________________________Label_2.Position = [383 427 111 22];
        app.Friction________________________________Label_2.Text = 'Minimum Depth';

        % Create Panel_18
        app.Panel_18 = uipanel(app.InputTab);
        app.Panel_18.AutoResizeChildren = 'off';
        app.Panel_18.Position = [10 501 351 84];

        % Create ButtonGroup_7
        app.ButtonGroup_7 = uibuttongroup(app.Panel_18);
```

```
            app.ButtonGroup_7.AutoResizeChildren = 'off';
            app.ButtonGroup_7.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_7SelectionChanged, true);
            app.ButtonGroup_7.ForegroundColor = [0.149 0.149 0.149];
            app.ButtonGroup_7.BorderType = 'none';
            app.ButtonGroup_7.Position = [25 9 100 64];

            % Create ImportDataButton
            app.ImportDataButton = uiradiobutton(app.ButtonGroup_7);
            app.ImportDataButton.Tooltip = {'Import the bathymetry.'; ''; 'Accepted formats: .txt, .tif'; ''; ' If a .tif file is uploaded, it will be automatically converted to a text file. In the conversion, the array is flipped up to down as FUNWAVE reads the depth file starting from the southwest corner.'};
            app.ImportDataButton.Text = ' Import Data';
            app.ImportDataButton.Position = [5 41 89 22];

            % Create FlatButton
            app.FlatButton = uiradiobutton(app.ButtonGroup_7);
            app.FlatButton.Tooltip = {'Create and use a flat bottom bathymetry'};
            app.FlatButton.Text = ' Flat';
            app.FlatButton.Position = [5 19 45 22];

            % Create SlopeButton
            app.SlopeButton = uiradiobutton(app.ButtonGroup_7);
            app.SlopeButton.Tooltip = {'Create and use a sloping bathymetry'};
            app.SlopeButton.Text = ' Slope';
            app.SlopeButton.Position = [5 -3 56 22];

            % Create Button4
            app.Button4 = uiradiobutton(app.ButtonGroup_7);
            app.Button4.Visible = 'off';
            app.Button4.Text = 'Button4';
            app.Button4.Position = [57 -4 27 22];
            app.Button4.Value = true;

            % Create DepthFileEditField
            app.DepthFileEditField = uieditfield(app.Panel_18, 'text');
            app.DepthFileEditField.ValueChangedFcn = createCallbackFcn(app, @DepthFileEditFieldValueChanged, true);
            app.DepthFileEditField.ValueChangingFcn = createCallbackFcn(app, @DepthFileEditFieldValueChanging, true);
            app.DepthFileEditField.Editable = 'off';
            app.DepthFileEditField.FontSize = 11;
            app.DepthFileEditField.FontColor = [0 0 1];
            app.DepthFileEditField.Visible = 'off';
            app.DepthFileEditField.Position = [133 52 144 17];

            % Create Button_8
            app.Button_8 = uibutton(app.Panel_18, 'push');
            app.Button_8.ButtonPushedFcn = createCallbackFcn(app, @Button_8Pushed, true);
            app.Button_8.VerticalAlignment = 'top';
            app.Button_8.FontSize = 10;
            app.Button_8.FontAngle = 'italic';
            app.Button_8.FontColor = [0 0.451 0.7412];
            app.Button_8.Visible = 'off';
            app.Button_8.Tooltip = {'Import the bathymetry.'; ''; 'Accepted formats: .txt, .tif'; ''; 'If a .tif file is uploaded, it will be automatically converted to a text file.'; ''; 'The dimensions and grid details are automatically extracted from the file.'; ''; 'The values in the .txt file should include decimal places.'};
            app.Button_8.Position = [285 51 19 18];
            app.Button_8.Text = '...';

            % Create WaterlevelmEditField
            app.WaterlevelmEditField = uieditfield(app.Panel_18, 'numeric');
            app.WaterlevelmEditField.Limits = [0 Inf];
            app.WaterlevelmEditField.ValueDisplayFormat = '%5.2f';
            app.WaterlevelmEditField.ValueChangedFcn = createCallbackFcn(app, @WaterlevelmEditFieldValueChanged, true);
            app.WaterlevelmEditField.FontColor = [0.651 0.651 0.651];
            app.WaterlevelmEditField.Visible = 'off';
            app.WaterlevelmEditField.Tooltip = {''};
            app.WaterlevelmEditField.Position = [233 9 44 17];

            % Create WaterlevelmEditFieldLabel
            app.WaterlevelmEditFieldLabel = uilabel(app.Panel_18);
            app.WaterlevelmEditFieldLabel.Visible = 'off';
            app.WaterlevelmEditFieldLabel.Tooltip = {'Adjust the initial water level of the input bathymetry.'; ''; 'For example, to add surge or tide effects.'};
            app.WaterlevelmEditFieldLabel.Position = [133 6 96 22];
            app.WaterlevelmEditFieldLabel.Text = 'Initial waver level';

            % Create InvertvaluesCheckBox
            app.InvertvaluesCheckBox = uicheckbox(app.Panel_18);
            app.InvertvaluesCheckBox.ValueChangedFcn = createCallbackFcn(app, @InvertvaluesCheckBoxValueChanged, true);
            app.InvertvaluesCheckBox.Visible = 'off';
            app.InvertvaluesCheckBox.Tooltip = {'Convert bathymetry values to positive and land values to negative'};
            app.InvertvaluesCheckBox.Text = 'Invert values';
            app.InvertvaluesCheckBox.WordWrap = 'on';
            app.InvertvaluesCheckBox.Position = [133 28 105 22];

            % Create DepthmEditField
            app.DepthmEditField = uieditfield(app.Panel_18, 'numeric');
            app.DepthmEditField.Limits = [0 Inf];
            app.DepthmEditField.ValueDisplayFormat = '%5.2f';
            app.DepthmEditField.ValueChangedFcn = createCallbackFcn(app, @DepthmEditFieldValueChanged, true);
            app.DepthmEditField.FontColor = [0.651 0.651 0.651];
            app.DepthmEditField.Visible = 'off';
            app.DepthmEditField.Tooltip = {''};
            app.DepthmEditField.Position = [190 31 48 17];
            app.DepthmEditField.Value = 100;

            % Create DepthmEditFieldLabel
            app.DepthmEditFieldLabel = uilabel(app.Panel_18);
            app.DepthmEditFieldLabel.HorizontalAlignment = 'right';
            app.DepthmEditFieldLabel.Visible = 'off';
            app.DepthmEditFieldLabel.Tooltip = {'Unit: meters'};
            app.DepthmEditFieldLabel.Position = [126 29 41 22];
            app.DepthmEditFieldLabel.Text = ' Depth';

            % Create StartatXEditFieldLabel
            app.StartatXEditFieldLabel = uilabel(app.Panel_18);
            app.StartatXEditFieldLabel.HorizontalAlignment = 'right';
            app.StartatXEditFieldLabel.Visible = 'off';
            app.StartatXEditFieldLabel.Tooltip = {'The location where the sloping beach begins, with a zero value indicating the deeper part of the water.'; ''; 'Unit: meters'};
            app.StartatXEditFieldLabel.Position = [127 6 56 22];
            app.StartatXEditFieldLabel.Text = 'Start at X';

            % Create StartatXEditField
            app.StartatXEditField = uieditfield(app.Panel_18, 'numeric');
            app.StartatXEditField.ValueDisplayFormat = '%3.2f';
            app.StartatXEditField.ValueChangedFcn = createCallbackFcn(app, @StartatXEditFieldValueChanged, true);
            app.StartatXEditField.FontColor = [0.651 0.651 0.651];
            app.StartatXEditField.Visible = 'off';
            app.StartatXEditField.Tooltip = {''};
            app.StartatXEditField.Position = [190 9 48 17];

            % Create AngleEditFieldLabel
            app.AngleEditFieldLabel = uilabel(app.Panel_18);
            app.AngleEditFieldLabel.HorizontalAlignment = 'right';
            app.AngleEditFieldLabel.Visible = 'off';
            app.AngleEditFieldLabel.Tooltip = {'Slope angle'};
            app.AngleEditFieldLabel.Position = [248 8 36 22];
            app.AngleEditFieldLabel.Text = 'Angle';

            % Create AngleEditField
            app.AngleEditField = uieditfield(app.Panel_18, 'numeric');
            app.AngleEditField.ValueDisplayFormat = '%3.2f';
            app.AngleEditField.ValueChangedFcn = createCallbackFcn(app, @AngleEditFieldValueChanged, true);
            app.AngleEditField.FontColor = [0.651 0.651 0.651];
            app.AngleEditField.Visible = 'off';
            app.AngleEditField.Tooltip = {''};
            app.AngleEditField.Position = [289 11 42 16];

            % Create Panel_19
            app.Panel_19 = uipanel(app.InputTab);
            app.Panel_19.AutoResizeChildren = 'off';
            app.Panel_19.Position = [10 454 351 36];

            % Create YEditField_2Label
            app.YEditField_2Label = uilabel(app.Panel_19);
            app.YEditField_2Label.HorizontalAlignment = 'right';
            app.YEditField_2Label.Tooltip = {'Matrix size in Y direction'};
            app.YEditField_2Label.Position = [182 6 16 22];
            app.YEditField_2Label.Text = 'Y';

            % Create YEditField_2
            app.YEditField_2 = uieditfield(app.Panel_19, 'numeric');
            app.YEditField_2.ValueDisplayFormat = '%.0f';
            app.YEditField_2.ValueChangedFcn = createCallbackFcn(app, @YEditField_2ValueChanged, true);
            app.YEditField_2.FontColor = [0.651 0.651 0.651];
            app.YEditField_2.Tooltip = {''};
            app.YEditField_2.Position = [204 9 45 16];

            % Create XEditField_2Label
            app.XEditField_2Label = uilabel(app.Panel_19);
            app.XEditField_2Label.HorizontalAlignment = 'right';
            app.XEditField_2Label.Tooltip = {'Matrix size in X direction'};
            app.XEditField_2Label.Position = [97 6 18 22];
            app.XEditField_2Label.Text = 'X ';

            % Create XEditField_2
            app.XEditField_2 = uieditfield(app.Panel_19, 'numeric');
            app.XEditField_2.Limits = [0 Inf];
            app.XEditField_2.ValueDisplayFormat = '%.0f';
            app.XEditField_2.ValueChangedFcn = createCallbackFcn(app, @XEditField_2ValueChanged, true);
            app.XEditField_2.FontColor = [0.651 0.651 0.651];
            app.XEditField_2.Tooltip = {''};
            app.XEditField_2.Position = [119 9 45 16];

            % Create Panel_20
            app.Panel_20 = uipanel(app.InputTab);
            app.Panel_20.AutoResizeChildren = 'off';
            app.Panel_20.Position = [10 403 351 39];

            % Create ResultsfolderEditFieldLabel
            app.ResultsfolderEditFieldLabel = uilabel(app.Panel_20);
            app.ResultsfolderEditFieldLabel.Tooltip = {'Folder where  the tsunami simulations will be saved'};
            app.ResultsfolderEditFieldLabel.Position = [53 6 79 22];
            app.ResultsfolderEditFieldLabel.Text = 'Results folder';

            % Create ResultsfolderEditField
            app.ResultsfolderEditField = uieditfield(app.Panel_20, 'text');
            app.ResultsfolderEditField.ValueChangedFcn = createCallbackFcn(app, @ResultsfolderEditFieldValueChanged, true);
            app.ResultsfolderEditField.ValueChangingFcn = createCallbackFcn(app, @ResultsfolderEditFieldValueChanging, true);
            app.ResultsfolderEditField.FontSize = 11;
            app.ResultsfolderEditField.FontColor = [0 0 1];
            app.ResultsfolderEditField.Tooltip = {''};
            app.ResultsfolderEditField.Placeholder = 'Default: "Simulation_Results/"';
            app.ResultsfolderEditField.Position = [139 9 178 16];

            % Create PrintLabel
            app.PrintLabel = uilabel(app.InputTab);
            app.PrintLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.PrintLabel.FontSize = 13;
            app.PrintLabel.FontWeight = 'bold';
            app.PrintLabel.FontColor = [0.0314 0.3686 0.6];
            app.PrintLabel.Position = [21 432 42 21];
            app.PrintLabel.Text = '  Print';

            % Create Panel_21
            app.Panel_21 = uipanel(app.InputTab);
            app.Panel_21.AutoResizeChildren = 'off';
            app.Panel_21.Position = [10 332 351 60];

            % Create ScreenLoadingIntervalLabel
            app.ScreenLoadingIntervalLabel = uilabel(app.Panel_21);
            app.ScreenLoadingIntervalLabel.WordWrap = 'on';
            app.ScreenLoadingIntervalLabel.Tooltip = {'Initila time'};
            app.ScreenLoadingIntervalLabel.Position = [55 27 31 22];
            app.ScreenLoadingIntervalLabel.Text = 'Start';

            % Create StartEditField
            app.StartEditField = uieditfield(app.Panel_21, 'numeric');
            app.StartEditField.Limits = [0 Inf];
            app.StartEditField.ValueDisplayFormat = '%.1f';
            app.StartEditField.ValueChangedFcn = createCallbackFcn(app, @StartEditFieldValueChanged, true);
            app.StartEditField.FontColor = [0.651 0.651 0.651];
            app.StartEditField.Tooltip = {''};
            app.StartEditField.Position = [90 30 52 16];

            % Create GaugeIntervalEditFieldLabel
            app.GaugeIntervalEditFieldLabel = uilabel(app.Panel_21);
            app.GaugeIntervalEditFieldLabel.HorizontalAlignment = 'right';
            app.GaugeIntervalEditFieldLabel.Tooltip = {'Time interval between station recordings.'; ''; 'Use 0.0 when the stations are for the nesting data.'; ''; 'Unit: seconds'};
            app.GaugeIntervalEditFieldLabel.Position = [154 3 110 28];
            app.GaugeIntervalEditFieldLabel.Text = 'Station Plot Interval';

            % Create GaugeIntervalEditField
            app.GaugeIntervalEditField = uieditfield(app.Panel_21, 'numeric');
            app.GaugeIntervalEditField.Limits = [0 Inf];
            app.GaugeIntervalEditField.ValueDisplayFormat = '%.1f';
            app.GaugeIntervalEditField.ValueChangedFcn = createCallbackFcn(app, @GaugeIntervalEditFieldValueChanged, true);
            app.GaugeIntervalEditField.FontColor = [0.651 0.651 0.651];
            app.GaugeIntervalEditField.Tooltip = {''};
            app.GaugeIntervalEditField.Position = [270 9 47 16];

            % Create SimulationIntervalEditFieldLabel
            app.SimulationIntervalEditFieldLabel = uilabel(app.Panel_21);
            app.SimulationIntervalEditFieldLabel.HorizontalAlignment = 'right';
            app.SimulationIntervalEditFieldLabel.WordWrap = 'on';
            app.SimulationIntervalEditFieldLabel.Tooltip = {'Interval for generating simulation output'; ''; 'Unit: seconds'};
            app.SimulationIntervalEditFieldLabel.Position = [154 29 110 18];
            app.SimulationIntervalEditFieldLabel.Text = 'Simulation Interval';

            % Create SimulationIntervalEditField
            app.SimulationIntervalEditField = uieditfield(app.Panel_21, 'numeric');
            app.SimulationIntervalEditField.Limits = [0 Inf];
            app.SimulationIntervalEditField.ValueDisplayFormat = '%.1f';
            app.SimulationIntervalEditField.ValueChangedFcn = createCallbackFcn(app, @SimulationIntervalEditFieldValueChanged, true);
            app.SimulationIntervalEditField.FontColor = [0.651 0.651 0.651];
            app.SimulationIntervalEditField.Tooltip = {''};
            app.SimulationIntervalEditField.Position = [270 30 47 16];

            % Create TotalEditFieldLabel
            app.TotalEditFieldLabel = uilabel(app.Panel_21);
            app.TotalEditFieldLabel.WordWrap = 'on';
            app.TotalEditFieldLabel.Tooltip = {'Total simulation time'; ''; 'Unit: seconds'};
            app.TotalEditFieldLabel.Position = [55 6 34 22];
            app.TotalEditFieldLabel.Text = 'Total';

            % Create TotalEditField
            app.TotalEditField = uieditfield(app.Panel_21, 'numeric');
            app.TotalEditField.Limits = [0 Inf];
            app.TotalEditField.ValueDisplayFormat = '%.1f';
            app.TotalEditField.ValueChangedFcn = createCallbackFcn(app, @TotalEditFieldValueChanged2, true);
            app.TotalEditField.FontColor = [0.651 0.651 0.651];
            app.TotalEditField.Tooltip = {''};
            app.TotalEditField.Position = [90 9 52 16];
            app.TotalEditField.Value = 3600;

            % Create TimeLabel_2
            app.TimeLabel_2 = uilabel(app.InputTab);
            app.TimeLabel_2.BackgroundColor = [0.9412 0.9412 0.9412];
            app.TimeLabel_2.FontSize = 13;
            app.TimeLabel_2.FontWeight = 'bold';
            app.TimeLabel_2.FontColor = [0.0314 0.3686 0.6];
            app.TimeLabel_2.Position = [22 381 47 21];
            app.TimeLabel_2.Text = '  Time';

            % Create Panel_22
            app.Panel_22 = uipanel(app.InputTab);
            app.Panel_22.AutoResizeChildren = 'off';
            app.Panel_22.Position = [10 214 351 107];

            % Create XEditFieldLabel
            app.XEditFieldLabel = uilabel(app.Panel_22);
            app.XEditFieldLabel.HorizontalAlignment = 'right';
            app.XEditFieldLabel.Tooltip = {''};
            app.XEditFieldLabel.Position = [214 29 25 22];
            app.XEditFieldLabel.Text = 'X ';

            % Create XEditField
            app.XEditField = uieditfield(app.Panel_22, 'numeric');
            app.XEditField.Limits = [0 Inf];
            app.XEditField.ValueDisplayFormat = '%8.5f';
            app.XEditField.ValueChangedFcn = createCallbackFcn(app, @XEditFieldValueChanged, true);
            app.XEditField.FontColor = [0.651 0.651 0.651];
            app.XEditField.Tooltip = {''};
            app.XEditField.Position = [245 33 57 16];

            % Create YgridsizeLabel
            app.YgridsizeLabel = uilabel(app.Panel_22);
            app.YgridsizeLabel.HorizontalAlignment = 'right';
            app.YgridsizeLabel.Position = [211 7 25 22];
            app.YgridsizeLabel.Text = 'Y';

            % Create YEditField
            app.YEditField = uieditfield(app.Panel_22, 'numeric');
            app.YEditField.Limits = [0 Inf];
            app.YEditField.ValueDisplayFormat = '%8.5f';
            app.YEditField.ValueChangedFcn = createCallbackFcn(app, ↙
@YEditFieldValueChanged, true);
            app.YEditField.FontColor = [0.651 0.651 0.651];
            app.YEditField.Tooltip = {''};
            app.YEditField.Position = [245 11 57 16];

            % Create LongitudeEditFieldLabel
            app.LongitudeEditFieldLabel = uilabel(app.Panel_22);
            app.LongitudeEditFieldLabel.WordWrap = 'on';
            app.LongitudeEditFieldLabel.Tooltip = {''};
            app.LongitudeEditFieldLabel.Position = [29 31 60 22];
            app.LongitudeEditFieldLabel.Text = 'Longitude';

            % Create LongitudeEditField
            app.LongitudeEditField = uieditfield(app.Panel_22, 'numeric');
            app.LongitudeEditField.ValueDisplayFormat = '%11.5f';
            app.LongitudeEditField.ValueChangedFcn = createCallbackFcn(app, ↙
@LongitudeEditFieldValueChanged, true);
            app.LongitudeEditField.FontColor = [0.651 0.651 0.651];
            app.LongitudeEditField.Enable = 'off';
            app.LongitudeEditField.Tooltip = {''};
            app.LongitudeEditField.Position = [88 34 76 16];

            % Create LatitudeEditFieldLabel
            app.LatitudeEditFieldLabel = uilabel(app.Panel_22);
            app.LatitudeEditFieldLabel.WordWrap = 'on';
            app.LatitudeEditFieldLabel.Tooltip = {''};
            app.LatitudeEditFieldLabel.Position = [29 6 46 28];
            app.LatitudeEditFieldLabel.Text = 'Latitude';

            % Create LatitudeEditField
            app.LatitudeEditField = uieditfield(app.Panel_22, 'numeric');
            app.LatitudeEditField.ValueDisplayFormat = '%11.5f';
            app.LatitudeEditField.ValueChangedFcn = createCallbackFcn(app, ↙
@LatitudeEditFieldValueChanged, true);
            app.LatitudeEditField.FontColor = [0.651 0.651 0.651];
            app.LatitudeEditField.Enable = 'off';
            app.LatitudeEditField.Tooltip = {''};
            app.LatitudeEditField.Position = [88 11 76 17];

            % Create ButtonGroup_4
            app.ButtonGroup_4 = uibuttongroup(app.Panel_22);
            app.ButtonGroup_4.AutoResizeChildren = 'off';
            app.ButtonGroup_4.SelectionChangedFcn = createCallbackFcn(app, ↙
@ButtonGroup_4SelectionChanged, true);
            app.ButtonGroup_4.Tooltip = {'Grid coordinate'};
            app.ButtonGroup_4.BorderType = 'none';
            app.ButtonGroup_4.Position = [82 69 187 30];

            % Create SphericalButton
            app.SphericalButton = uiradiobutton(app.ButtonGroup_4);
            app.SphericalButton.Tooltip = {'Unit: degrees'};
            app.SphericalButton.Text = ' Spherical';
            app.SphericalButton.Position = [11 7 76 22];

            % Create CartesianButton
            app.CartesianButton = uiradiobutton(app.ButtonGroup_4);
            app.CartesianButton.Tooltip = {'Unit: meters'};
            app.CartesianButton.Text = ' Cartesian';
            app.CartesianButton.FontColor = [0.149 0.149 0.149];
            app.CartesianButton.Position = [99 7 76 22];

            % Create Button_2
            app.Button_2 = uiradiobutton(app.ButtonGroup_4);
            app.Button_2.Visible = 'off';
            app.Button_2.Text = '';
            app.Button_2.Position = [181 7 25 22];
            app.Button_2.Value = true;

            % Create SouthwestcornerLabel
            app.SouthwestcornerLabel = uilabel(app.Panel_22);
            app.SouthwestcornerLabel.Tooltip = {'The corner coordinates of the numerical grid.'; ''; 'Unit: degrees / meters'};
            app.SouthwestcornerLabel.Position = [28 50 104 22];
            app.SouthwestcornerLabel.Text = 'Southwest corner:';

            % Create GridSizeLabel
            app.GridSizeLabel = uilabel(app.Panel_22);
            app.GridSizeLabel.Tooltip = {'Spatial resolution of the numerical grid.'; ''; 'Unit: meters'};
            app.GridSizeLabel.Position = [227 50 58 22];
            app.GridSizeLabel.Text = 'Grid Size:';

            % Create GridLabel
            app.GridLabel = uilabel(app.InputTab);
            app.GridLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.GridLabel.FontSize = 13;
            app.GridLabel.FontWeight = 'bold';
            app.GridLabel.FontColor = [0.0314 0.3686 0.6];
            app.GridLabel.Position = [22 310 46 21];
            app.GridLabel.Text = '  Grid  ';

            % Create Panel_23
            app.Panel_23 = uipanel(app.InputTab);
            app.Panel_23.AutoResizeChildren = 'off';
            app.Panel_23.Position = [10 59 351 144];

            % Create LayerwidthLabel
            app.LayerwidthLabel = uilabel(app.Panel_23);
            app.LayerwidthLabel.Tooltip = {'Sponge layer thickness relative to model boundary. The width should be at least half of the wavelength.'; ''; 'Unit: meters'; ''; 'Example: '; 'Entering 10,000 in the ''North'' textbox extends the sponge layer 10,000 meters southward from the North boundary.'};
            app.LayerwidthLabel.Position = [140 46 71 23];
            app.LayerwidthLabel.Text = 'Layer width:';

            % Create DirectCheckBox
            app.DirectCheckBox = uicheckbox(app.Panel_23);
            app.DirectCheckBox.ValueChangedFcn = createCallbackFcn(app, @DirectCheckBoxValueChanged, true);
            app.DirectCheckBox.Tooltip = {'L-D type sponge'};
            app.DirectCheckBox.Text = ' Direct';
            app.DirectCheckBox.Position = [29 107 57 22];
            app.DirectCheckBox.Value = true;

            % Create FrictionCheckBox
            app.FrictionCheckBox = uicheckbox(app.Panel_23);
            app.FrictionCheckBox.ValueChangedFcn = createCallbackFcn(app, @FrictionCheckBoxValueChanged, true);
            app.FrictionCheckBox.Text = ' Friction';
            app.FrictionCheckBox.Position = [138 107 65 22];

            % Create DiffusionCheckBox
            app.DiffusionCheckBox = uicheckbox(app.Panel_23);
            app.DiffusionCheckBox.ValueChangedFcn = createCallbackFcn(app, @DiffusionCheckBoxValueChanged, true);
            app.DiffusionCheckBox.Text = ' Diffusion';
            app.DiffusionCheckBox.Position = [255 107 72 22];

            % Create NorthEditField
            app.NorthEditField = uieditfield(app.Panel_23, 'numeric');
            app.NorthEditField.Limits = [0 Inf];
            app.NorthEditField.ValueDisplayFormat = '%5.1f';
            app.NorthEditField.ValueChangedFcn = createCallbackFcn(app, @NorthEditFieldValueChanged, true);
            app.NorthEditField.FontColor = [0.651 0.651 0.651];
            app.NorthEditField.Tooltip = {''};
            app.NorthEditField.Position = [93 29 61 16];
            app.NorthEditField.Value = 100000;

            % Create EastEditField
            app.EastEditField = uieditfield(app.Panel_23, 'numeric');
            app.EastEditField.Limits = [0 Inf];
            app.EastEditField.ValueDisplayFormat = '%5.1f';
            app.EastEditField.ValueChangedFcn = createCallbackFcn(app, @EastEditFieldValueChanged, true);
            app.EastEditField.FontColor = [0.651 0.651 0.651];
            app.EastEditField.Tooltip = {''};
            app.EastEditField.Position = [218 29 62 16];
            app.EastEditField.Value = 100000;

            % Create SouthEditField
            app.SouthEditField = uieditfield(app.Panel_23, 'numeric');
            app.SouthEditField.Limits = [0 Inf];
            app.SouthEditField.ValueDisplayFormat = '%5.1f';
            app.SouthEditField.ValueChangedFcn = createCallbackFcn(app, @SouthEditFieldValueChanged, true);
            app.SouthEditField.FontColor = [0.651 0.651 0.651];
            app.SouthEditField.Tooltip = {''};
            app.SouthEditField.Position = [93 8 61 16];
            app.SouthEditField.Value = 100000;

            % Create WestEditField
            app.WestEditField = uieditfield(app.Panel_23, 'numeric');
            app.WestEditField.Limits = [0 Inf];
            app.WestEditField.ValueDisplayFormat = '%5.1f';
            app.WestEditField.ValueChangedFcn = createCallbackFcn(app, @WestEditFieldValueChanged, true);
            app.WestEditField.FontColor = [0.651 0.651 0.651];
            app.WestEditField.Tooltip = {''};
            app.WestEditField.Position = [218 8 62 16];
            app.WestEditField.Value = 100000;

            % Create NorthLabel
            app.NorthLabel = uilabel(app.Panel_23);
            app.NorthLabel.Tooltip = {'Unit: meters'};
            app.NorthLabel.Position = [53 26 35 22];
            app.NorthLabel.Text = 'North';

            % Create EastLabel
            app.EastLabel = uilabel(app.Panel_23);
            app.EastLabel.Tooltip = {'Unit: meters'};
            app.EastLabel.Position = [185 26 29 22];
            app.EastLabel.Text = 'East';

            % Create WestLabel
            app.WestLabel = uilabel(app.Panel_23);
            app.WestLabel.Tooltip = {'Unit: meters'};
            app.WestLabel.Position = [185 5 32 22];
            app.WestLabel.Text = 'West';

            % Create SouthLabel
            app.SouthLabel = uilabel(app.Panel_23);
            app.SouthLabel.Tooltip = {'Unit: meters'};
            app.SouthLabel.Position = [53 5 37 22];
            app.SouthLabel.Text = 'South';

            % Create DecayTypeDropDownLabel
            app.DecayTypeDropDownLabel = uilabel(app.Panel_23);
            app.DecayTypeDropDownLabel.Position = [29 87 66 22];
            app.DecayTypeDropDownLabel.Text = 'Decay type';

            % Create DecayTypeDropDown
            app.DecayTypeDropDown = uidropdown(app.Panel_23);
            app.DecayTypeDropDown.Items = {'Very mild', 'Mild', 'Sharp'};
            app.DecayTypeDropDown.ValueChangedFcn = createCallbackFcn(app, ↙
@DecayTypeDropDownValueChanged, true);
            app.DecayTypeDropDown.Position = [29 73 72 16];
            app.DecayTypeDropDown.Value = 'Sharp';

            % Create MaximumCdEditFieldLabel
            app.MaximumCdEditFieldLabel = uilabel(app.Panel_23);
            app.MaximumCdEditFieldLabel.Visible = 'off';
            app.MaximumCdEditFieldLabel.Tooltip = {'Cd = drag coefficient'};
            app.MaximumCdEditFieldLabel.Position = [138 87 78 22];
            app.MaximumCdEditFieldLabel.Text = 'Maximum Cd';

            % Create MaximumCdEditField
            app.MaximumCdEditField = uieditfield(app.Panel_23, 'numeric');
            app.MaximumCdEditField.Limits = [0 Inf];
            app.MaximumCdEditField.ValueDisplayFormat = '%11.3f';
            app.MaximumCdEditField.ValueChangedFcn = createCallbackFcn(app, ↙
@MaximumCdEditFieldValueChanged, true);
            app.MaximumCdEditField.FontColor = [0.651 0.651 0.651];
            app.MaximumCdEditField.Visible = 'off';
            app.MaximumCdEditField.Tooltip = {''};
            app.MaximumCdEditField.Position = [138 73 51 16];
            app.MaximumCdEditField.Value = 10;

            % Create MaximumCspEditFieldLabel
            app.MaximumCspEditFieldLabel = uilabel(app.Panel_23);
            app.MaximumCspEditFieldLabel.Visible = 'off';
            app.MaximumCspEditFieldLabel.Tooltip = {'Csd = maximum diffusion coefficient'};
            app.MaximumCspEditFieldLabel.Position = [255 86 84 22];
            app.MaximumCspEditFieldLabel.Text = 'Maximum Csp';

            % Create MaximumCspEditField
            app.MaximumCspEditField = uieditfield(app.Panel_23, 'numeric');
            app.MaximumCspEditField.Limits = [0 Inf];
            app.MaximumCspEditField.ValueDisplayFormat = '%11.3f';
            app.MaximumCspEditField.ValueChangedFcn = createCallbackFcn(app, @MaximumCspEditFieldValueChanged, true);
            app.MaximumCspEditField.FontColor = [0.651 0.651 0.651];
            app.MaximumCspEditField.Visible = 'off';
            app.MaximumCspEditField.Tooltip = {''};
            app.MaximumCspEditField.Position = [255 73 42 16];
            app.MaximumCspEditField.Value = 1;

            % Create SpongeBoundaryLabel
            app.SpongeBoundaryLabel = uilabel(app.InputTab);
            app.SpongeBoundaryLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.SpongeBoundaryLabel.FontSize = 13;
            app.SpongeBoundaryLabel.FontWeight = 'bold';
            app.SpongeBoundaryLabel.FontColor = [0.0314 0.3686 0.6];
            app.SpongeBoundaryLabel.Tooltip = {'Use sponge layer for wave damping'};
            app.SpongeBoundaryLabel.Position = [22 192 130 21];
            app.SpongeBoundaryLabel.Text = '  Sponge Boundary  ';

            % Create Panel_24
            app.Panel_24 = uipanel(app.InputTab);
            app.Panel_24.AutoResizeChildren = 'off';
            app.Panel_24.Position = [10 10 351 39];

            % Create ObstacleLoadFile
            app.ObstacleLoadFile = uibutton(app.Panel_24, 'push');
            app.ObstacleLoadFile.ButtonPushedFcn = createCallbackFcn(app, @Button_19Pushed, true);
            app.ObstacleLoadFile.VerticalAlignment = 'top';
            app.ObstacleLoadFile.Visible = 'off';
            app.ObstacleLoadFile.Tooltip = {''};
            app.ObstacleLoadFile.Position = [287 7 19 19];
            app.ObstacleLoadFile.Text = '...';

            % Create ObstacleFilename
            app.ObstacleFilename = uieditfield(app.Panel_24, 'text');
            app.ObstacleFilename.Editable = 'off';
            app.ObstacleFilename.FontSize = 11;
            app.ObstacleFilename.Visible = 'off';
            app.ObstacleFilename.Position = [161 8 123 16];

            % Create AddobstaclesCheckBox
            app.AddobstaclesCheckBox = uicheckbox(app.Panel_24);
            app.AddobstaclesCheckBox.ValueChangedFcn = createCallbackFcn(app, @AddobstaclesCheckBoxValueChanged, true);
            app.AddobstaclesCheckBox.Tooltip = {'Integrate obstacles with a file format matching the depth data: 1 represents water, and 0 represents dry points.'; ''; 'File size must be the same as the input bathymetry.'};
            app.AddobstaclesCheckBox.Text = ' Add obstacles';
            app.AddobstaclesCheckBox.Position = [52 5 102 22];

            % Create ObstaclesLabel
            app.ObstaclesLabel = uilabel(app.InputTab);
            app.ObstaclesLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.ObstaclesLabel.FontSize = 13;
            app.ObstaclesLabel.FontWeight = 'bold';
            app.ObstaclesLabel.FontColor = [0.0314 0.3686 0.6];
            app.ObstaclesLabel.Position = [22 37 78 21];
            app.ObstaclesLabel.Text = '  Obstacles  ';

            % Create PhysicsLabel
            app.PhysicsLabel = uilabel(app.InputTab);
            app.PhysicsLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.PhysicsLabel.HorizontalAlignment = 'center';
            app.PhysicsLabel.FontSize = 13;
            app.PhysicsLabel.FontWeight = 'bold';
            app.PhysicsLabel.FontColor = [0.0314 0.3686 0.6];
            app.PhysicsLabel.Position = [383 627 73 21];
            app.PhysicsLabel.Text = '  Physics   ';

            % Create Panel_26
            app.Panel_26 = uipanel(app.InputTab);
            app.Panel_26.AutoResizeChildren = 'off';
            app.Panel_26.Position = [371 450 356 60];

            % Create ButtonGroup_8
            app.ButtonGroup_8 = uibuttongroup(app.Panel_26);
            app.ButtonGroup_8.AutoResizeChildren = 'off';
            app.ButtonGroup_8.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_8SelectionChanged, true);
            app.ButtonGroup_8.BorderType = 'none';
            app.ButtonGroup_8.Position = [18 9 159 41];

            % Create ConstantCoefficientButton
            app.ConstantCoefficientButton = uiradiobutton(app.ButtonGroup_8);
            app.ConstantCoefficientButton.Tooltip = {'Set a constant drag friction value throughout the domain'};
            app.ConstantCoefficientButton.Text = ' Constant Coefficient';
            app.ConstantCoefficientButton.Position = [16 20 135 22];
            app.ConstantCoefficientButton.Value = true;

            % Create NonconstantButton
            app.NonconstantButton = uiradiobutton(app.ButtonGroup_8);
            app.NonconstantButton.Tooltip = {'Assign a non-uniform drag friction value across the domain.'};
            app.NonconstantButton.Text = ' Nonconstant';
            app.NonconstantButton.Position = [16 0 94 22];

            % Create FrictionFileEditFieldLabel
            app.FrictionFileEditFieldLabel = uilabel(app.Panel_26);
            app.FrictionFileEditFieldLabel.HorizontalAlignment = 'right';
            app.FrictionFileEditFieldLabel.Visible = 'off';
            app.FrictionFileEditFieldLabel.Tooltip = {'Import the friction file.'; ''; 'Dimension should be the same with the input bathymetry.'};
            app.FrictionFileEditFieldLabel.Position = [189 9 25 22];
            app.FrictionFileEditFieldLabel.Text = 'File';

            % Create FrictionFileEditField
            app.FrictionFileEditField = uieditfield(app.Panel_26, 'text');
            app.FrictionFileEditField.ValueChangingFcn = createCallbackFcn(app, @FrictionFileEditFieldValueChanging, true);
            app.FrictionFileEditField.Editable = 'off';
            app.FrictionFileEditField.FontSize = 11;
            app.FrictionFileEditField.Visible = 'off';
            app.FrictionFileEditField.Tooltip = {'Dimension should be the same with the input bathymetry'};
            app.FrictionFileEditField.Position = [222 12 84 16];

            % Create Button_10
            app.Button_10 = uibutton(app.Panel_26, 'push');
            app.Button_10.ButtonPushedFcn = createCallbackFcn(app, @Button_10Pushed, true);
            app.Button_10.VerticalAlignment = 'top';
            app.Button_10.Visible = 'off';
            app.Button_10.Tooltip = {''};
            app.Button_10.Position = [311 11 19 19];
            app.Button_10.Text = '...';

            % Create ValueEditFieldLabel
            app.ValueEditFieldLabel = uilabel(app.Panel_26);
            app.ValueEditFieldLabel.HorizontalAlignment = 'right';
            app.ValueEditFieldLabel.Tooltip = {'Set to 0 for no friction'};
            app.ValueEditFieldLabel.Position = [180 29 34 22];
            app.ValueEditFieldLabel.Text = 'Value';

            % Create ConstantFrictionValue
            app.ConstantFrictionValue = uieditfield(app.Panel_26, 'numeric');
            app.ConstantFrictionValue.ValueDisplayFormat = '%8.3f';
            app.ConstantFrictionValue.ValueChangedFcn = createCallbackFcn(app, @ConstantFrictionValueValueChanged, true);
            app.ConstantFrictionValue.FontColor = [0.651 0.651 0.651];
            app.ConstantFrictionValue.Tooltip = {''};
            app.ConstantFrictionValue.Position = [222 32 41 16];

            % Create BottomFrictionLabel
            app.BottomFrictionLabel = uilabel(app.InputTab);
            app.BottomFrictionLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.BottomFrictionLabel.FontSize = 13;
            app.BottomFrictionLabel.FontWeight = 'bold';
            app.BottomFrictionLabel.FontColor = [0.0314 0.3686 0.6];
            app.BottomFrictionLabel.Tooltip = {''};
            app.BottomFrictionLabel.Position = [383 500 114 21];
            app.BottomFrictionLabel.Text = '  Bottom Friction  ';

            % Create Panel_28
            app.Panel_28 = uipanel(app.InputTab);
            app.Panel_28.AutoResizeChildren = 'off';
            app.Panel_28.Position = [371 108 356 148];

            % Create CreatenestedgridboundaryCheckBox
            app.CreatenestedgridboundaryCheckBox = uicheckbox(app.Panel_28);
            app.CreatenestedgridboundaryCheckBox.ValueChangedFcn = createCallbackFcn(app, @CreatenestedgridboundaryCheckBoxValueChanged, true);
            app.CreatenestedgridboundaryCheckBox.Tooltip = {'Use gauges as points to define the boundary of the subgrid layer'};
            app.CreatenestedgridboundaryCheckBox.Text = 'Create nested grid boundary';
            app.CreatenestedgridboundaryCheckBox.WordWrap = 'on';
            app.CreatenestedgridboundaryCheckBox.Position = [33 84 182 23];

            % Create Button_7
            app.Button_7 = uibutton(app.Panel_28, 'push');
            app.Button_7.ButtonPushedFcn = createCallbackFcn(app, @Button_7Pushed2, true);
            app.Button_7.Enable = 'off';
            app.Button_7.Tooltip = {'Import the file containing a list of coordinates representing the gauge locations.'; ''; 'Text file format: Tab-delimited, with columns for latitude and longitude.'};
            app.Button_7.Position = [230 114 19 19];
            app.Button_7.Text = '...';

            % Create LoadFileEditField
            app.LoadFileEditField = uieditfield(app.Panel_28, 'text');
            app.LoadFileEditField.ValueChangedFcn = createCallbackFcn(app, @LoadFileEditFieldValueChanged, true);
            app.LoadFileEditField.ValueChangingFcn = createCallbackFcn(app, @LoadFileEditFieldValueChanging, true);
            app.LoadFileEditField.Editable = 'off';
            app.LoadFileEditField.FontSize = 11;
            app.LoadFileEditField.Enable = 'off';
            app.LoadFileEditField.Placeholder = '.shp or .txt';
            app.LoadFileEditField.Position = [152 115 74 16];

            % Create ImportgaugelistCheckBox
            app.ImportgaugelistCheckBox = uicheckbox(app.Panel_28);
            app.ImportgaugelistCheckBox.ValueChangedFcn = createCallbackFcn(app, @ImportgaugelistCheckBoxValueChanged, true);
            app.ImportgaugelistCheckBox.Tooltip = {'Add stations at specific locations to record wave patterns throughout the simulation'};
            app.ImportgaugelistCheckBox.Text = 'Import gauge list';
            app.ImportgaugelistCheckBox.Position = [33 112 112 22];

            % Create NorthEditField_2
            app.NorthEditField_2 = uieditfield(app.Panel_28, 'numeric');
            app.NorthEditField_2.ValueDisplayFormat = '%5.4f';
            app.NorthEditField_2.ValueChangedFcn = createCallbackFcn(app, @NorthEditField_2ValueChanged, true);
            app.NorthEditField_2.FontColor = [0.651 0.651 0.651];
            app.NorthEditField_2.Enable = 'off';
            app.NorthEditField_2.Tooltip = {'North limit'};
            app.NorthEditField_2.Position = [147 37 64 16];

            % Create EastEditField_2
            app.EastEditField_2 = uieditfield(app.Panel_28, 'numeric');
            app.EastEditField_2.ValueDisplayFormat = '%5.4f';
            app.EastEditField_2.ValueChangedFcn = createCallbackFcn(app, @EastEditField_2ValueChanged, true);
            app.EastEditField_2.FontColor = [0.651 0.651 0.651];
            app.EastEditField_2.Enable = 'off';
            app.EastEditField_2.Tooltip = {'East limit'};
            app.EastEditField_2.Position = [217 24 63 16];

            % Create SouthEditField_2
            app.SouthEditField_2 = uieditfield(app.Panel_28, 'numeric');
            app.SouthEditField_2.ValueDisplayFormat = '%5.4f';
            app.SouthEditField_2.ValueChangedFcn = createCallbackFcn(app, @SouthEditField_2ValueChanged, true);
            app.SouthEditField_2.FontColor = [0.651 0.651 0.651];
            app.SouthEditField_2.Enable = 'off';
            app.SouthEditField_2.Tooltip = {'South limit'};
            app.SouthEditField_2.Position = [147 13 64 16];

            % Create WestEditField_2
            app.WestEditField_2 = uieditfield(app.Panel_28, 'numeric');
            app.WestEditField_2.ValueDisplayFormat = '%5.4f';
            app.WestEditField_2.ValueChangedFcn = createCallbackFcn(app, @WestEditField_2ValueChanged, true);
            app.WestEditField_2.FontColor = [0.651 0.651 0.651];
            app.WestEditField_2.Enable = 'off';
            app.WestEditField_2.Tooltip = {'West limit'};
            app.WestEditField_2.Position = [79 24 63 16];

            % Create BoundarylimitsButton
            app.BoundarylimitsButton = uibutton(app.Panel_28, 'push');
            app.BoundarylimitsButton.ButtonPushedFcn = createCallbackFcn(app, @BoundarylimitsButtonPushed, true);
            app.BoundarylimitsButton.Enable = 'off';
            app.BoundarylimitsButton.Tooltip = {'Extract coordinate and grid spacing information from a tif file. '};
            app.BoundarylimitsButton.Position = [79 58 201 22];
            app.BoundarylimitsButton.Text = 'Boundary limits:';

            % Create SpacingEditFieldLabel
            app.SpacingEditFieldLabel = uilabel(app.Panel_28);
            app.SpacingEditFieldLabel.WordWrap = 'on';
            app.SpacingEditFieldLabel.Enable = 'off';
            app.SpacingEditFieldLabel.Tooltip = {'Set the gauge spacing'};
            app.SpacingEditFieldLabel.Position = [216 84 77 22];
            app.SpacingEditFieldLabel.Text = 'Grid spacing:';

            % Create SpacingEditField
            app.SpacingEditField = uieditfield(app.Panel_28, 'numeric');
            app.SpacingEditField.ValueDisplayFormat = '%5.4f';
            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 = [294 87 48 16];

            % Create Friction________________________________Label_5
            app.Friction________________________________Label_5 = uilabel(app.InputTab);
            app.Friction________________________________Label_5.BackgroundColor = [0.9412 0.9412 0.9412];
            app.Friction________________________________Label_5.FontSize = 13;
            app.Friction________________________________Label_5.FontWeight = 'bold';
            app.Friction________________________________Label_5.FontColor = [0.0314 0.3686 0.6];
            app.Friction________________________________Label_5.Position =
[383 246 65 21];
            app.Friction________________________________Label_5.Text = 'Gauges  ';

            % Create Panel_29
            app.Panel_29 = uipanel(app.InputTab);
            app.Panel_29.AutoResizeChildren = 'off';
            app.Panel_29.Position = [371 60 356 37];

            % Create Button_14
            app.Button_14 = uibutton(app.Panel_29, 'push');
            app.Button_14.ButtonPushedFcn = createCallbackFcn(app, @Button_14Pushed, true);
            app.Button_14.VerticalAlignment = 'top';
            app.Button_14.FontSize = 10;
            app.Button_14.FontAngle = 'italic';
            app.Button_14.Tooltip = {''};
            app.Button_14.Position = [288 7 19 18];
            app.Button_14.Text = '...';

            % Create SavefilestoLabel
            app.SavefilestoLabel = uilabel(app.Panel_29);
            app.SavefilestoLabel.Tooltip = {'Specify the directory where ''INPUT_FILES'', containing all generated files, will be saved.'};
            app.SavefilestoLabel.Position = [37 5 74 22];
            app.SavefilestoLabel.Text = 'Save files to:';

            % Create SavefilestoEditField
            app.SavefilestoEditField = uieditfield(app.Panel_29, 'text');
            app.SavefilestoEditField.ValueChangedFcn = createCallbackFcn(app, @SavefilestoEditFieldValueChanged, true);
            app.SavefilestoEditField.ValueChangingFcn = createCallbackFcn(app, @SavefilestoEditFieldValueChanging, true);
            app.SavefilestoEditField.Editable = 'off';
            app.SavefilestoEditField.FontSize = 11;
            app.SavefilestoEditField.FontColor = [0 0 1];
            app.SavefilestoEditField.Tooltip = {'Specify the directory where ''OUTPUT_FILES'' that contians all the files generated in this app will be saved'};
            app.SavefilestoEditField.Placeholder = 'Default: Desktop';
            app.SavefilestoEditField.Position = [113 8 170 16];

            % Create Friction________________________________Label_6
            app.Friction________________________________Label_6 = uilabel(app.InputTab);
            app.Friction________________________________Label_6.BackgroundColor = [0.9412 0.9412 0.9412];
            app.Friction________________________________Label_6.FontSize = 13;
            app.Friction________________________________Label_6.FontWeight = 'bold';
            app.Friction________________________________Label_6.FontColor = [0.0314 0.3686 0.6];
            app.Friction________________________________Label_6.Position = [383 87 73 21];
            app.Friction________________________________Label_6.Text = 'Directory  ';

            % Create DepthLabel
            app.DepthLabel = uilabel(app.InputTab);
            app.DepthLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.DepthLabel.FontSize = 13;
            app.DepthLabel.FontWeight = 'bold';
            app.DepthLabel.FontColor = [0.0314 0.3686 0.6];
            app.DepthLabel.Position = [21 574 52 21];
            app.DepthLabel.Text = '  Depth  ';

            % Create DimensionLabel
            app.DimensionLabel = uilabel(app.InputTab);
            app.DimensionLabel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.DimensionLabel.FontSize = 13;
            app.DimensionLabel.FontWeight = 'bold';
            app.DimensionLabel.FontColor = [0.0314 0.3686 0.6];
            app.DimensionLabel.Position = [21 479 86 21];
            app.DimensionLabel.Text = '  Dimension  ';

            % Create Panel_30
            app.Panel_30 = uipanel(app.InputTab);
            app.Panel_30.AutoResizeChildren = 'off';
            app.Panel_30.Position = [371 329 356 55];

            % Create ButtonGroup_23
            app.ButtonGroup_23 = uibuttongroup(app.Panel_30);
            app.ButtonGroup_23.AutoResizeChildren = 'off';
            app.ButtonGroup_23.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_23SelectionChanged, true);
            app.ButtonGroup_23.BorderType = 'none';
            app.ButtonGroup_23.Position = [26 2 155 42];

            % Create ShockwavecapturingButton
            app.ShockwavecapturingButton = uiradiobutton(app.ButtonGroup_23);
            app.ShockwavecapturingButton.Enable = 'off';
            app.ShockwavecapturingButton.Text = ' Shock wave capturing';
            app.ShockwavecapturingButton.Position = [7 22 145 22];
            app.ShockwavecapturingButton.Value = true;

            % Create ViscositybreakingButton
            app.ViscositybreakingButton = uiradiobutton(app.ButtonGroup_23);
            app.ViscositybreakingButton.Enable = 'off';
            app.ViscositybreakingButton.Text = ' Viscosity breaking';
            app.ViscositybreakingButton.Position = [7 4 122 22];

            % Create C1EditField
            app.C1EditField = uieditfield(app.Panel_30, 'numeric');
            app.C1EditField.Limits = [0 Inf];
            app.C1EditField.ValueDisplayFormat = '%.2f';
            app.C1EditField.ValueChangedFcn = createCallbackFcn(app, @C1EditFieldValueChanged, true);
            app.C1EditField.FontColor = [0.651 0.651 0.651];
            app.C1EditField.Visible = 'off';
            app.C1EditField.Tooltip = {''};
            app.C1EditField.Position = [222 27 41 16];
            app.C1EditField.Value = 0.65;

            % Create C1EditFieldLabel
            app.C1EditFieldLabel = uilabel(app.Panel_30);
            app.C1EditFieldLabel.HorizontalAlignment = 'right';
            app.C1EditFieldLabel.Visible = 'off';
            app.C1EditFieldLabel.Tooltip = {'Parameter value from Kennedy et↙
al. (2000)'};
            app.C1EditFieldLabel.Position = [192 23 25 22];
            app.C1EditFieldLabel.Text = 'C1';

            % Create C2EditFieldLabel
            app.C2EditFieldLabel = uilabel(app.Panel_30);
            app.C2EditFieldLabel.HorizontalAlignment = 'right';
            app.C2EditFieldLabel.Visible = 'off';
            app.C2EditFieldLabel.Tooltip = {'Parameter value from Kennedy et↙
al. (2000)'};
            app.C2EditFieldLabel.Position = [192 4 25 22];
            app.C2EditFieldLabel.Text = 'C2';

            % Create C2EditField
            app.C2EditField = uieditfield(app.Panel_30, 'numeric');
            app.C2EditField.Limits = [0 Inf];
            app.C2EditField.ValueDisplayFormat = '%.2f';
            app.C2EditField.ValueChangedFcn = createCallbackFcn(app, ↙
@C2EditFieldValueChanged, true);
            app.C2EditField.FontColor = [0.651 0.651 0.651];
            app.C2EditField.Visible = 'off';
            app.C2EditField.Tooltip = {''};
            app.C2EditField.Position = [222 8 41 16];
            app.C2EditField.Value = 0.35;

            % Create Friction________________________________Label_7
            app.Friction________________________________Label_7 = uilabel(app.↙
InputTab);
            app.Friction________________________________Label_7.↙
BackgroundColor = [0.9412 0.9412 0.9412];
            app.Friction________________________________Label_7.FontSize = 13;
            app.Friction________________________________Label_7.FontWeight =↙
'bold';
            app.Friction________________________________Label_7.FontColor =↙
[0.0314 0.3686 0.6];
            app.Friction________________________________Label_7.Position =↙
[383 373 109 22];
            app.Friction________________________________Label_7.Text = '  Wave↙
Breaking ';

            % Create GenerateinputtxtButton
            app.GenerateinputtxtButton = uibutton(app.InputTab, 'push');
            app.GenerateinputtxtButton.ButtonPushedFcn = createCallbackFcn↙
(app, @GenerateinputtxtButtonPushed, true);
            app.GenerateinputtxtButton.BackgroundColor = [0.9216 0.9216↙
0.9216];
            app.GenerateinputtxtButton.FontSize = 15;
            app.GenerateinputtxtButton.FontWeight = 'bold';
            app.GenerateinputtxtButton.Position = [370 10 357 39];
            app.GenerateinputtxtButton.Text = 'Generate input.txt';

            % Create Panel_33
            app.Panel_33 = uipanel(app.InputTab);
            app.Panel_33.AutoResizeChildren = 'off';
            app.Panel_33.Position = [371 268 356 50];

            % Create UseSmagorinskyCheckBox
            app.UseSmagorinskyCheckBox = uicheckbox(app.Panel_33);
            app.UseSmagorinskyCheckBox.ValueChangedFcn = createCallbackFcn(app, @UseSmagorinskyCheckBoxValueChanged2, true);
            app.UseSmagorinskyCheckBox.Tooltip = {'Apply a Smagorinsky-like method for subgrid turbulent mixing'};
            app.UseSmagorinskyCheckBox.Text = ' Smagorinsky mixing';
            app.UseSmagorinskyCheckBox.Position = [33 12 134 22];

            % Create TimeintervalsecEditFieldLabel
            app.TimeintervalsecEditFieldLabel = uilabel(app.Panel_33);
            app.TimeintervalsecEditFieldLabel.Visible = 'off';
            app.TimeintervalsecEditFieldLabel.Tooltip = {'Set time interval for calculating values'; ''; 'Unit: seconds'};
            app.TimeintervalsecEditFieldLabel.Position = [196 23 74 22];
            app.TimeintervalsecEditFieldLabel.Text = 'Time interval';

            % Create TimeintervalsecEditField
            app.TimeintervalsecEditField = uieditfield(app.Panel_33, 'numeric');
            app.TimeintervalsecEditField.Limits = [0 Inf];
            app.TimeintervalsecEditField.FontColor = [0.651 0.651 0.651];
            app.TimeintervalsecEditField.Visible = 'off';
            app.TimeintervalsecEditField.Position = [273 26 41 16];
            app.TimeintervalsecEditField.Value = 20;

            % Create SteadyTimeEditFieldLabel
            app.SteadyTimeEditFieldLabel = uilabel(app.Panel_33);
            app.SteadyTimeEditFieldLabel.Visible = 'off';
            app.SteadyTimeEditFieldLabel.Tooltip = {'Set the initial time for calculating mean values'; ''; 'Unit: seconds'};
            app.SteadyTimeEditFieldLabel.Position = [196 3 70 22];
            app.SteadyTimeEditFieldLabel.Text = 'Steady time';

            % Create SteadyTimeEditField
            app.SteadyTimeEditField = uieditfield(app.Panel_33, 'numeric');
            app.SteadyTimeEditField.Limits = [0 Inf];
            app.SteadyTimeEditField.FontColor = [0.651 0.651 0.651];
            app.SteadyTimeEditField.Visible = 'off';
            app.SteadyTimeEditField.Position = [273 6 41 16];

            % Create Friction________________________________Label_10
            app.Friction________________________________Label_10 = uilabel(app.InputTab);
            app.Friction________________________________Label_10.BackgroundColor = [0.9412 0.9412 0.9412];
            app.Friction________________________________Label_10.FontSize = 13;
            app.Friction________________________________Label_10.FontWeight = 'bold';
            app.Friction________________________________Label_10.FontColor = [0.0314 0.3686 0.6];
            app.Friction________________________________Label_10.Position = [383 306 116 22];
            app.Friction________________________________Label_10.Text = ' Turbulent Mixing';

            % Create InitialConditionTab
            app.InitialConditionTab = uitab(app.TabGroup);
            app.InitialConditionTab.AutoResizeChildren = 'off';
            app.InitialConditionTab.Title = 'Initial Condition';
            app.InitialConditionTab.BackgroundColor = [0.9412 0.9412 0.9412];
            app.InitialConditionTab.Interruptible = 'off';

            % Create TabGroup2
            app.TabGroup2 = uitabgroup(app.InitialConditionTab);
            app.TabGroup2.AutoResizeChildren = 'off';
            app.TabGroup2.TabLocation = 'left';
            app.TabGroup2.SelectionChangedFcn = createCallbackFcn(app, @TabGroup2SelectionChanged, true);
            app.TabGroup2.Position = [-69 84 806 515];

            % Create Tab_5
            app.Tab_5 = uitab(app.TabGroup2);
            app.Tab_5.AutoResizeChildren = 'off';

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

            % Create Label_9
            app.Label_9 = uilabel(app.Tab);
            app.Label_9.BackgroundColor = [0.651 0.651 0.651];
            app.Label_9.Position = [1 495 738 22];
            app.Label_9.Text = '';

            % Create ImportUVZfilesLabel
            app.ImportUVZfilesLabel = uilabel(app.Tab);
            app.ImportUVZfilesLabel.FontSize = 14;
            app.ImportUVZfilesLabel.FontWeight = 'bold';
            app.ImportUVZfilesLabel.FontColor = [1 1 1];
            app.ImportUVZfilesLabel.Position = [24 495 131 22];
            app.ImportUVZfilesLabel.Text = 'Import  U, V, Z files';

            % Create Button_32
            app.Button_32 = uibutton(app.Tab, 'push');
            app.Button_32.ButtonPushedFcn = createCallbackFcn(app, @Button_4Pushed, true);
            app.Button_32.VerticalAlignment = 'top';
            app.Button_32.Tooltip = {'Import the Z file. '; ''; 'Accepted format: .txt'; ''; 'If you only have .tif file, convert this file to .txt in ''Input'' tab > Depth > Import Data'; ''; 'The values in the .txt file should include decimal places.'};
            app.Button_32.Position = [523 371 22 22];
            app.Button_32.Text = '...';

            % Create Button_34
            app.Button_34 = uibutton(app.Tab, 'push');
            app.Button_34.ButtonPushedFcn = createCallbackFcn(app, @Button_5Pushed, true);
            app.Button_34.Tooltip = {'Import the U vector file. '; ''; 'Accepted format: .txt'; ''; 'If you only have .tif file, convert this file to .txt in ''Input'' tab > Depth > Import Data'; ''; 'The values in the .txt file should include decimal places.'};
            app.Button_34.Position = [523 339 22 22];
            app.Button_34.Text = '...';

            % Create SurfaceHeightEditFieldLabel
            app.SurfaceHeightEditFieldLabel = uilabel(app.Tab);
            app.SurfaceHeightEditFieldLabel.HorizontalAlignment = 'right';
            app.SurfaceHeightEditFieldLabel.Tooltip = {'Import file containing the sea surface displacement matrix.'; ''; 'The values in the .txt file should include decimal places.'};
            app.SurfaceHeightEditFieldLabel.Position = [196 371 85 22];
            app.SurfaceHeightEditFieldLabel.Text = 'Surface Height';

            % Create SurfaceHeightEditField
            app.SurfaceHeightEditField = uieditfield(app.Tab, 'text');
            app.SurfaceHeightEditField.ValueChangingFcn = createCallbackFcn(app, @SurfaceHeightEditFieldValueChanging, true);
            app.SurfaceHeightEditField.Editable = 'off';
            app.SurfaceHeightEditField.FontSize = 11;
            app.SurfaceHeightEditField.Tooltip = {''};
            app.SurfaceHeightEditField.Position = [307 371 207 22];

            % Create UVelocityEditFieldLabel
            app.UVelocityEditFieldLabel = uilabel(app.Tab);
            app.UVelocityEditFieldLabel.HorizontalAlignment = 'right';
            app.UVelocityEditFieldLabel.Tooltip = {'Import file with x-direction velocity matrix.'; ''; 'The values in the .txt file should include decimal places.'};
            app.UVelocityEditFieldLabel.Position = [196 339 59 22];
            app.UVelocityEditFieldLabel.Text = 'U Velocity';

            % Create UVelocityEditField
            app.UVelocityEditField = uieditfield(app.Tab, 'text');
            app.UVelocityEditField.ValueChangingFcn = createCallbackFcn(app, @UVelocityEditFieldValueChanging, true);
            app.UVelocityEditField.Editable = 'off';
            app.UVelocityEditField.FontSize = 11;
            app.UVelocityEditField.Tooltip = {''};
            app.UVelocityEditField.Position = [307 339 207 22];

            % Create VVelocityEditFieldLabel
            app.VVelocityEditFieldLabel = uilabel(app.Tab);
            app.VVelocityEditFieldLabel.HorizontalAlignment = 'right';
            app.VVelocityEditFieldLabel.Tooltip = {'Import file with y-direction velocity matrix.'; ''; 'The values in the .txt file should include decimal places.'};
            app.VVelocityEditFieldLabel.Position = [197 307 58 22];
            app.VVelocityEditFieldLabel.Text = 'V Velocity';

            % Create VVelocityEditField
            app.VVelocityEditField = uieditfield(app.Tab, 'text');
            app.VVelocityEditField.ValueChangingFcn = createCallbackFcn(app, @VVelocityEditFieldValueChanging, true);
            app.VVelocityEditField.Editable = 'off';
            app.VVelocityEditField.FontSize = 11;
            app.VVelocityEditField.Tooltip = {''};
            app.VVelocityEditField.Position = [307 306 207 22];

            % Create Button_33
            app.Button_33 = uibutton(app.Tab, 'push');
            app.Button_33.ButtonPushedFcn = createCallbackFcn(app, @Button 6Pushed, true);
```

```matlab
            app.Button_33.Tooltip = {'Import the V vector file. '; ''; 'Accepted format: .txt'; ''; 'If you only have .tif file, convert this file to .txt in ''Input'' tab > Depth > Import Data'; ''; 'The values in the .txt file should include decimal places.'};
            app.Button_33.Position = [523 306 22 22];
            app.Button_33.Text = '...';

            % Create Tab_2
            app.Tab_2 = uitab(app.TabGroup2);
            app.Tab_2.AutoResizeChildren = 'off';

            % Create Label_10
            app.Label_10 = uilabel(app.Tab_2);
            app.Label_10.BackgroundColor = [0.651 0.651 0.651];
            app.Label_10.Position = [1 495 751 22];
            app.Label_10.Text = '';

            % Create WavemakerParametersLabel_2
            app.WavemakerParametersLabel_2 = uilabel(app.Tab_2);
            app.WavemakerParametersLabel_2.FontSize = 14;
            app.WavemakerParametersLabel_2.FontWeight = 'bold';
            app.WavemakerParametersLabel_2.FontColor = [1 1 1];
            app.WavemakerParametersLabel_2.Position = [24 495 165 22];
            app.WavemakerParametersLabel_2.Text = 'Wavemaker Parameters';

            % Create Button_20
            app.Button_20 = uibutton(app.Tab_2, 'push');
            app.Button_20.ButtonPushedFcn = createCallbackFcn(app, @Button_20Pushed, true);
            app.Button_20.Enable = 'off';
            app.Button_20.Tooltip = {''};
            app.Button_20.Position = [404 77 22 18];
            app.Button_20.Text = '...';

            % Create WavemakertypeDropDownLabel
            app.WavemakertypeDropDownLabel = uilabel(app.Tab_2);
            app.WavemakertypeDropDownLabel.HorizontalAlignment = 'right';
            app.WavemakertypeDropDownLabel.Position = [161 459 95 22];
            app.WavemakertypeDropDownLabel.Text = 'Wavemaker type';

            % Create WavemakertypeDropDown
            app.WavemakertypeDropDown = uidropdown(app.Tab_2);
            app.WavemakertypeDropDown.Items = {'See options', 'INI_REC', 'INI_GAUSSIAN', 'LEF_SOL', 'INI_SOL', 'JON_1D', 'JON_2D', 'WK_IRR', 'WK_REG', 'WK_TIME_SERIES', 'WK_DATA2D', 'TMA_1D'};
            app.WavemakertypeDropDown.DropDownOpeningFcn = createCallbackFcn(app, @WavemakertypeDropDownOpening, true);
            app.WavemakertypeDropDown.ValueChangedFcn = createCallbackFcn(app, @WavemakertypeDropDownValueChanged, true);
            app.WavemakertypeDropDown.Position = [310 460 134 22];
            app.WavemakertypeDropDown.Value = 'See options';

            % Create AmplitudemEditFieldLabel
            app.AmplitudemEditFieldLabel = uilabel(app.Tab_2);
            app.AmplitudemEditFieldLabel.HorizontalAlignment = 'right';
            app.AmplitudemEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.AmplitudemEditFieldLabel.Tooltip = {'Height of the initial sea surface displacement'};
            app.AmplitudemEditFieldLabel.Position = [161 427 80 22];
            app.AmplitudemEditFieldLabel.Text = 'Amplitude (m)';

            % Create AmplitudemEditField
            app.AmplitudemEditField = uieditfield(app.Tab_2, 'numeric');
            app.AmplitudemEditField.Limits = [0 Inf];
            app.AmplitudemEditField.ValueDisplayFormat = '%11.2f';
            app.AmplitudemEditField.ValueChangedFcn = createCallbackFcn(app, @AmplitudemEditFieldValueChanged, true);
            app.AmplitudemEditField.FontColor = [0.8 0.8 0.8];
            app.AmplitudemEditField.Enable = 'off';
            app.AmplitudemEditField.Tooltip = {''};
            app.AmplitudemEditField.Position = [310 429 70 18];

            % Create WaterdepthmEditFieldLabel
            app.WaterdepthmEditFieldLabel = uilabel(app.Tab_2);
            app.WaterdepthmEditFieldLabel.HorizontalAlignment = 'right';
            app.WaterdepthmEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.WaterdepthmEditFieldLabel.Tooltip = {'Water depth at the location of the wavemaker'};
            app.WaterdepthmEditFieldLabel.Position = [161 395 91 22];
            app.WaterdepthmEditFieldLabel.Text = 'Water depth (m)';

            % Create WaterdepthmEditField
            app.WaterdepthmEditField = uieditfield(app.Tab_2, 'numeric');
            app.WaterdepthmEditField.Limits = [0 Inf];
            app.WaterdepthmEditField.ValueDisplayFormat = '%11.2f';
            app.WaterdepthmEditField.ValueChangedFcn = createCallbackFcn(app, @WaterdepthmEditFieldValueChanged, true);
            app.WaterdepthmEditField.FontColor = [0.8 0.8 0.8];
            app.WaterdepthmEditField.Enable = 'off';
            app.WaterdepthmEditField.Tooltip = {''};
            app.WaterdepthmEditField.Position = [310 397 70 18];

            % Create LagtimesecEditFieldLabel
            app.LagtimesecEditFieldLabel = uilabel(app.Tab_2);
            app.LagtimesecEditFieldLabel.HorizontalAlignment = 'right';
            app.LagtimesecEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.LagtimesecEditFieldLabel.Tooltip = {'Lag time associated with the solitary wave generated at the left boundary'};
            app.LagtimesecEditFieldLabel.Position = [161 139 80 22];
            app.LagtimesecEditFieldLabel.Text = 'Lag time (sec)';

            % Create LagtimesecEditField
            app.LagtimesecEditField = uieditfield(app.Tab_2, 'numeric');
            app.LagtimesecEditField.Limits = [0 Inf];
            app.LagtimesecEditField.ValueDisplayFormat = '%.2f';
            app.LagtimesecEditField.ValueChangedFcn = createCallbackFcn(app, @LagtimesecEditFieldValueChanged, true);
            app.LagtimesecEditField.FontColor = [0.8 0.8 0.8];
            app.LagtimesecEditField.Enable = 'off';
            app.LagtimesecEditField.Tooltip = {''};
            app.LagtimesecEditField.Position = [310 141 70 18];

            % Create XcoordinatemEditFieldLabel
            app.XcoordinatemEditFieldLabel = uilabel(app.Tab_2);
            app.XcoordinatemEditFieldLabel.HorizontalAlignment = 'right';
            app.XcoordinatemEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.XcoordinatemEditFieldLabel.Tooltip = {'Coordinate in X direction.'; ''; 'For INI_REC, it refers to the location of the center of the rectangular hump'};
            app.XcoordinatemEditFieldLabel.Position = [161 363 93 22];
            app.XcoordinatemEditFieldLabel.Text = 'X coordinate (m)';

            % Create XcoordinatemEditField
            app.XcoordinatemEditField = uieditfield(app.Tab_2, 'numeric');
            app.XcoordinatemEditField.ValueDisplayFormat = '%.4f';
            app.XcoordinatemEditField.ValueChangedFcn = createCallbackFcn(app, @XcoordinatemEditFieldValueChanged, true);
            app.XcoordinatemEditField.FontColor = [0.8 0.8 0.8];
            app.XcoordinatemEditField.Enable = 'off';
            app.XcoordinatemEditField.Tooltip = {''};
            app.XcoordinatemEditField.Position = [310 365 70 18];

            % Create YcoordinatemEditFieldLabel
            app.YcoordinatemEditFieldLabel = uilabel(app.Tab_2);
            app.YcoordinatemEditFieldLabel.HorizontalAlignment = 'right';
            app.YcoordinatemEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.YcoordinatemEditFieldLabel.Tooltip = {'Coordinate in Y direction'; ''; 'For INI_REC, it refers to the location of the center of the rectangular hump'};
            app.YcoordinatemEditFieldLabel.Position = [161 331 94 22];
            app.YcoordinatemEditFieldLabel.Text = 'Y coordinate (m)';

            % Create YcoordinatemEditField
            app.YcoordinatemEditField = uieditfield(app.Tab_2, 'numeric');
            app.YcoordinatemEditField.ValueDisplayFormat = '%.4f';
            app.YcoordinatemEditField.ValueChangedFcn = createCallbackFcn(app, @YcoordinatemEditFieldValueChanged, true);
            app.YcoordinatemEditField.FontColor = [0.8 0.8 0.8];
            app.YcoordinatemEditField.Enable = 'off';
            app.YcoordinatemEditField.Tooltip = {''};
            app.YcoordinatemEditField.Position = [310 333 70 18];

            % Create WidthmEditFieldLabel
            app.WidthmEditFieldLabel = uilabel(app.Tab_2);
            app.WidthmEditFieldLabel.HorizontalAlignment = 'right';
            app.WidthmEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.WidthmEditFieldLabel.Tooltip = {'For INI_REC or INI_GAUSSIAN, it refers to the width of the rectangular hump '; ''; 'For WK_REG, it is the width in Y direction'};
            app.WidthmEditFieldLabel.Position = [161 299 57 22];
            app.WidthmEditFieldLabel.Text = 'Width (m)';

            % Create WidthmEditField
            app.WidthmEditField = uieditfield(app.Tab_2, 'numeric');
            app.WidthmEditField.ValueDisplayFormat = '%.2f';
            app.WidthmEditField.ValueChangedFcn = createCallbackFcn(app, @WidthmEditFieldValueChanged, true);
            app.WidthmEditField.FontColor = [0.8 0.8 0.8];
            app.WidthmEditField.Enable = 'off';
            app.WidthmEditField.Tooltip = {''};
            app.WidthmEditField.Position = [310 301 70 18];

            % Create TimerampsecEditFieldLabel
            app.TimerampsecEditFieldLabel = uilabel(app.Tab_2);
            app.TimerampsecEditFieldLabel.HorizontalAlignment = 'right';
            app.TimerampsecEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.TimerampsecEditFieldLabel.Tooltip = {'The period during which wave height gradually increases from zero to its maximum'};
            app.TimerampsecEditFieldLabel.Position = [161 203 91 22];
            app.TimerampsecEditFieldLabel.Text = 'Time ramp (sec)';

            % Create TimerampsecEditField
            app.TimerampsecEditField = uieditfield(app.Tab_2, 'numeric');
            app.TimerampsecEditField.Limits = [0 Inf];
            app.TimerampsecEditField.ValueDisplayFormat = '%.2f';
            app.TimerampsecEditField.ValueChangedFcn = createCallbackFcn(app, @TimerampsecEditFieldValueChanged, true);
            app.TimerampsecEditField.FontColor = [0.8 0.8 0.8];
            app.TimerampsecEditField.Enable = 'off';
            app.TimerampsecEditField.Tooltip = {''};
            app.TimerampsecEditField.Position = [310 205 70 18];

            % Create WidthDeltaEditFieldLabel
            app.WidthDeltaEditFieldLabel = uilabel(app.Tab_2);
            app.WidthDeltaEditFieldLabel.HorizontalAlignment = 'right';
            app.WidthDeltaEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.WidthDeltaEditFieldLabel.Tooltip = {'Delta width parameter'; ''; 'Default values: 0.3 ~ 0.6'};
            app.WidthDeltaEditFieldLabel.Position = [161 267 68 22];
            app.WidthDeltaEditFieldLabel.Text = 'Width Delta';

            % Create WidthDeltaEditField
            app.WidthDeltaEditField = uieditfield(app.Tab_2, 'numeric');
            app.WidthDeltaEditField.ValueDisplayFormat = '%.2f';
            app.WidthDeltaEditField.ValueChangedFcn = createCallbackFcn(app, @WidthDeltaEditFieldValueChanged, true);
            app.WidthDeltaEditField.FontColor = [0.8 0.8 0.8];
            app.WidthDeltaEditField.Enable = 'off';
            app.WidthDeltaEditField.Tooltip = {''};
            app.WidthDeltaEditField.Position = [310 269 70 18];

            % Create PeriodsecEditFieldLabel
            app.PeriodsecEditFieldLabel = uilabel(app.Tab_2);
            app.PeriodsecEditFieldLabel.HorizontalAlignment = 'right';
            app.PeriodsecEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.PeriodsecEditFieldLabel.Tooltip = {'Time period of the regular wave'};
            app.PeriodsecEditFieldLabel.Position = [161 171 69 22];
            app.PeriodsecEditFieldLabel.Text = 'Period (sec)';

            % Create PeriodsecEditField
            app.PeriodsecEditField = uieditfield(app.Tab_2, 'numeric');
            app.PeriodsecEditField.Limits = [0 Inf];
            app.PeriodsecEditField.ValueDisplayFormat = '%.2f';
            app.PeriodsecEditField.ValueChangedFcn = createCallbackFcn(app, @PeriodsecEditFieldValueChanged, true);
            app.PeriodsecEditField.FontColor = [0.8 0.8 0.8];
            app.PeriodsecEditField.Enable = 'off';
            app.PeriodsecEditField.Tooltip = {''};
            app.PeriodsecEditField.Position = [310 173 70 18];

            % Create ThetadegreesEditFieldLabel
```

```matlab
            app.ThetadegreesEditFieldLabel = uilabel(app.Tab_2);
            app.ThetadegreesEditFieldLabel.HorizontalAlignment = 'right';
            app.ThetadegreesEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.ThetadegreesEditFieldLabel.Tooltip = {'Direction of regular wave'};
            app.ThetadegreesEditFieldLabel.Position = [161 235 88 22];
            app.ThetadegreesEditFieldLabel.Text = 'Theta (degrees)';

            % Create ThetadegreesEditField
            app.ThetadegreesEditField = uieditfield(app.Tab_2, 'numeric');
            app.ThetadegreesEditField.ValueDisplayFormat = '%.2f';
            app.ThetadegreesEditField.ValueChangedFcn = createCallbackFcn(app, @ThetadegreesEditFieldValueChanged, true);
            app.ThetadegreesEditField.FontColor = [0.8 0.8 0.8];
            app.ThetadegreesEditField.Enable = 'off';
            app.ThetadegreesEditField.Tooltip = {''};
            app.ThetadegreesEditField.Position = [310 237 70 18];

            % Create WavecomponentfileEditFieldLabel
            app.WavecomponentfileEditFieldLabel = uilabel(app.Tab_2);
            app.WavecomponentfileEditFieldLabel.HorizontalAlignment = 'right';
            app.WavecomponentfileEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.WavecomponentfileEditFieldLabel.Position = [161 75 118 22];
            app.WavecomponentfileEditFieldLabel.Text = 'Wave component file';

            % Create WavecomponentfileEditField
            app.WavecomponentfileEditField = uieditfield(app.Tab_2, 'text');
            app.WavecomponentfileEditField.ValueChangedFcn = createCallbackFcn(app, @WavecomponentfileEditFieldValueChanged, true);
            app.WavecomponentfileEditField.Editable = 'off';
            app.WavecomponentfileEditField.FontColor = [0.8 0.8 0.8];
            app.WavecomponentfileEditField.Enable = 'off';
            app.WavecomponentfileEditField.Tooltip = {'Import the file that contains the wave component'};
            app.WavecomponentfileEditField.Position = [311 77 89 18];

            % Create WavecomponentcountEditFieldLabel
            app.WavecomponentcountEditFieldLabel = uilabel(app.Tab_2);
            app.WavecomponentcountEditFieldLabel.HorizontalAlignment = 'right';
            app.WavecomponentcountEditFieldLabel.FontColor = [0.149 0.149 0.149];
            app.WavecomponentcountEditFieldLabel.Tooltip = {'Number of the wave components'};
            app.WavecomponentcountEditFieldLabel.Position = [161 107 133 22];
            app.WavecomponentcountEditFieldLabel.Text = 'Wave component count';

            % Create WaveCompCountEditField
            app.WaveCompCountEditField = uieditfield(app.Tab_2, 'numeric');
            app.WaveCompCountEditField.ValueDisplayFormat = '%.2f';
            app.WaveCompCountEditField.FontColor = [0.8 0.8 0.8];
            app.WaveCompCountEditField.Enable = 'off';
            app.WaveCompCountEditField.Tooltip = {''};
            app.WaveCompCountEditField.Position = [311 109 70 18];

            % Create FrequencyLabel
            app.FrequencyLabel = uilabel(app.Tab_2);
            app.FrequencyLabel.Position = [166 13 62 22];
            app.FrequencyLabel.Text = 'Frequency';

            % Create FrequencyPeakLabel
            app.FrequencyPeakLabel = uilabel(app.Tab_2);
            app.FrequencyPeakLabel.HorizontalAlignment = 'right';
            app.FrequencyPeakLabel.Position = [550 13 32 22];
            app.FrequencyPeakLabel.Text = 'Peak';

            % Create PeakEditField
            app.PeakEditField = uieditfield(app.Tab_2, 'numeric');
            app.PeakEditField.ValueDisplayFormat = '%11.2f';
            app.PeakEditField.ValueChangedFcn = createCallbackFcn(app, @PeakEditFieldValueChanged, true);
            app.PeakEditField.FontColor = [0.8 0.8 0.8];
            app.PeakEditField.Enable = 'off';
            app.PeakEditField.Position = [588 15 48 18];

            % Create MinimumEditFieldLabel
            app.MinimumEditFieldLabel = uilabel(app.Tab_2);
            app.MinimumEditFieldLabel.HorizontalAlignment = 'right';
            app.MinimumEditFieldLabel.Position = [307 13 55 22];
            app.MinimumEditFieldLabel.Text = 'Minimum';

            % Create MinimumEditField
            app.MinimumEditField = uieditfield(app.Tab_2, 'numeric');
            app.MinimumEditField.ValueDisplayFormat = '%11.2f';
            app.MinimumEditField.ValueChangedFcn = createCallbackFcn(app, @MinimumEditFieldValueChanged, true);
            app.MinimumEditField.FontColor = [0.8 0.8 0.8];
            app.MinimumEditField.Enable = 'off';
            app.MinimumEditField.Position = [369 15 48 18];

            % Create MaximumEditFieldLabel
            app.MaximumEditFieldLabel = uilabel(app.Tab_2);
            app.MaximumEditFieldLabel.HorizontalAlignment = 'right';
            app.MaximumEditFieldLabel.Position = [427 13 58 22];
            app.MaximumEditFieldLabel.Text = 'Maximum';

            % Create MaximumEditField
            app.MaximumEditField = uieditfield(app.Tab_2, 'numeric');
            app.MaximumEditField.ValueDisplayFormat = '%11.2f';
            app.MaximumEditField.ValueChangedFcn = createCallbackFcn(app, @MaximumEditFieldValueChanged, true);
            app.MaximumEditField.FontColor = [0.8 0.8 0.8];
            app.MaximumEditField.Enable = 'off';
            app.MaximumEditField.Position = [491 14 48 18];

            % Create WaveHeightmEditFieldLabel
            app.WaveHeightmEditFieldLabel = uilabel(app.Tab_2);
            app.WaveHeightmEditFieldLabel.HorizontalAlignment = 'right';
            app.WaveHeightmEditFieldLabel.Tooltip = {'Significant wave height in Wei and Kirby (1999)'};
            app.WaveHeightmEditFieldLabel.Position = [161 44 93 22];
            app.WaveHeightmEditFieldLabel.Text = 'Wave Height (m)';

            % Create WaveHeightmEditField
            app.WaveHeightmEditField = uieditfield(app.Tab_2, 'numeric');
            app.WaveHeightmEditField.ValueDisplayFormat = '%11.2f';
            app.WaveHeightmEditField.ValueChangedFcn = createCallbackFcn(app, @WaveHeightmEditFieldValueChanged, true);
            app.WaveHeightmEditField.FontColor = [0.8 0.8 0.8];
            app.WaveHeightmEditField.Enable = 'off';
            app.WaveHeightmEditField.Position = [311 46 48 18];

            % Create Tab_4
            app.Tab_4 = uitab(app.TabGroup2);
            app.Tab_4.AutoResizeChildren = 'off';

            % Create Label_11
            app.Label_11 = uilabel(app.Tab_4);
            app.Label_11.BackgroundColor = [0.651 0.651 0.651];
            app.Label_11.Position = [1 495 738 22];
            app.Label_11.Text = '';

            % Create MeteotsunamiLabel
            app.MeteotsunamiLabel = uilabel(app.Tab_4);
            app.MeteotsunamiLabel.FontSize = 14;
            app.MeteotsunamiLabel.FontWeight = 'bold';
            app.MeteotsunamiLabel.FontColor = [1 1 1];
            app.MeteotsunamiLabel.Position = [24 495 101 22];
            app.MeteotsunamiLabel.Text = 'Meteotsunami';

            % Create Button_29
            app.Button_29 = uibutton(app.Tab_4, 'push');
            app.Button_29.ButtonPushedFcn = createCallbackFcn(app, @Button_29Pushed, true);
            app.Button_29.Enable = 'off';
            app.Button_29.Tooltip = {''};
            app.Button_29.Position = [575 345 22 20];
            app.Button_29.Text = '...';

            % Create Button_30
            app.Button_30 = uibutton(app.Tab_4, 'push');
            app.Button_30.ButtonPushedFcn = createCallbackFcn(app, @Button_30Pushed, true);
            app.Button_30.Enable = 'off';
            app.Button_30.Tooltip = {''};
            app.Button_30.Position = [575 226 22 20];
            app.Button_30.Text = '...';

            % Create WindforceSwitchLabel
            app.WindforceSwitchLabel = uilabel(app.Tab_4);
            app.WindforceSwitchLabel.HorizontalAlignment = 'center';
            app.WindforceSwitchLabel.Tooltip = {'Use wind effect'};
            app.WindforceSwitchLabel.Position = [81 303 64 22];
            app.WindforceSwitchLabel.Text = 'Wind force';

            % Create WindforceSwitch
            app.WindforceSwitch = uiswitch(app.Tab_4, 'slider');
            app.WindforceSwitch.ValueChangedFcn = createCallbackFcn(app, @WindforceSwitchValueChanged, true);
            app.WindforceSwitch.Tooltip = {''};
            app.WindforceSwitch.Position = [243 307 32 14];

```

```
        % Create AirpressureSwitchLabel
        app.AirpressureSwitchLabel = uilabel(app.Tab_4);
        app.AirpressureSwitchLabel.HorizontalAlignment = 'center';
        app.AirpressureSwitchLabel.Position = [81 429 70 22];
        app.AirpressureSwitchLabel.Text = 'Air pressure';

        % Create AirpressureSwitch
        app.AirpressureSwitch = uiswitch(app.Tab_4, 'slider');
        app.AirpressureSwitch.ValueChangedFcn = createCallbackFcn(app, @AirpressureSwitchValueChanged, true);
        app.AirpressureSwitch.Tooltip = {'Use air pressure effect'};
        app.AirpressureSwitch.Position = [243 433 32 14];
        app.AirpressureSwitch.Value = 'On';

        % Create WindstresscoefficientEditField
        app.WindstresscoefficientEditField = uieditfield(app.Tab_4, 'numeric');
        app.WindstresscoefficientEditField.Limits = [0 Inf];
        app.WindstresscoefficientEditField.ValueDisplayFormat = '%11.3g';
        app.WindstresscoefficientEditField.ValueChangedFcn = createCallbackFcn(app, @WindstresscoefficientEditFieldValueChanged, true);
        app.WindstresscoefficientEditField.FontColor = [0.651 0.651 0.651];
        app.WindstresscoefficientEditField.Enable = 'off';
        app.WindstresscoefficientEditField.Tooltip = {''};
        app.WindstresscoefficientEditField.Position = [518 305 50 18];
        app.WindstresscoefficientEditField.Value = 0.002;

        % Create WindstresscoefficientEditFieldLabel
        app.WindstresscoefficientEditFieldLabel = uilabel(app.Tab_4);
        app.WindstresscoefficientEditFieldLabel.HorizontalAlignment = 'right';
        app.WindstresscoefficientEditFieldLabel.Enable = 'off';
        app.WindstresscoefficientEditFieldLabel.Tooltip = {'Value used in the quadratic formula'};
        app.WindstresscoefficientEditFieldLabel.Position = [341 303 127 22];
        app.WindstresscoefficientEditFieldLabel.Text = 'Wind stress coefficient';

        % Create CrestRatioEditFieldLabel
        app.CrestRatioEditFieldLabel = uilabel(app.Tab_4);
        app.CrestRatioEditFieldLabel.HorizontalAlignment = 'right';
        app.CrestRatioEditFieldLabel.Enable = 'off';
        app.CrestRatioEditFieldLabel.Tooltip = {'The ratio of the forced wave crest height to the maximum surface elevation.'; ''; 'For storm surges, default value is 1.'};
        app.CrestRatioEditFieldLabel.Position = [341 264 140 22];
        app.CrestRatioEditFieldLabel.Text = 'Wind crest percent (ratio)';

        % Create CrestRatioEditField
        app.CrestRatioEditField = uieditfield(app.Tab_4, 'numeric');
        app.CrestRatioEditField.Limits = [0 Inf];
        app.CrestRatioEditField.ValueDisplayFormat = '%11.2f';
        app.CrestRatioEditField.ValueChangedFcn = createCallbackFcn(app, @CrestRatioEditFieldValueChanged, true);
        app.CrestRatioEditField.FontColor = [0.651 0.651 0.651];
        app.CrestRatioEditField.Enable = 'off';
```

```
            app.CrestRatioEditField.Tooltip = {'Ratio of the wave crest height↙
to the maximum surface height. '; ''; 'Default for storm surge: 1.0'};
            app.CrestRatioEditField.Position = [519 266 49 18];
            app.CrestRatioEditField.Value = 1;

            % Create ConstantwindfieldSwitchLabel
            app.ConstantwindfieldSwitchLabel = uilabel(app.Tab_4);
            app.ConstantwindfieldSwitchLabel.HorizontalAlignment = 'center';
            app.ConstantwindfieldSwitchLabel.Tooltip = {'Use a constant wind↙
field'};
            app.ConstantwindfieldSwitchLabel.Position = [81 343 109 22];
            app.ConstantwindfieldSwitchLabel.Text = 'Constant wind field';

            % Create ConstantwindfieldSwitch
            app.ConstantwindfieldSwitch = uiswitch(app.Tab_4, 'slider');
            app.ConstantwindfieldSwitch.ValueChangedFcn = createCallbackFcn↙
(app, @ConstantwindfieldSwitchValueChanged, true);
            app.ConstantwindfieldSwitch.Tooltip = {''};
            app.ConstantwindfieldSwitch.Position = [243 347 32 15];

            % Create ConstantwindfileEditFieldLabel
            app.ConstantwindfileEditFieldLabel = uilabel(app.Tab_4);
            app.ConstantwindfileEditFieldLabel.HorizontalAlignment = 'right';
            app.ConstantwindfileEditFieldLabel.Enable = 'off';
            app.ConstantwindfileEditFieldLabel.Position = [341 344 102 22];
            app.ConstantwindfileEditFieldLabel.Text = 'Constant wind file';

            % Create ConstantwindfileEditField
            app.ConstantwindfileEditField = uieditfield(app.Tab_4, 'text');
            app.ConstantwindfileEditField.Editable = 'off';
            app.ConstantwindfileEditField.Enable = 'off';
            app.ConstantwindfileEditField.Tooltip = {'Import the wind file'};
            app.ConstantwindfileEditField.Position = [453 345 115 20];

            % Create HollandstormmodelSwitchLabel
            app.HollandstormmodelSwitchLabel = uilabel(app.Tab_4);
            app.HollandstormmodelSwitchLabel.HorizontalAlignment = 'center';
            app.HollandstormmodelSwitchLabel.Tooltip = {'Use the Holland↙
model'};
            app.HollandstormmodelSwitchLabel.Position = [81 225 118 22];
            app.HollandstormmodelSwitchLabel.Text = 'Holland storm model';

            % Create HollandstormmodelSwitch
            app.HollandstormmodelSwitch = uiswitch(app.Tab_4, 'slider');
            app.HollandstormmodelSwitch.ValueChangedFcn = createCallbackFcn↙
(app, @HollandstormmodelSwitchValueChanged, true);
            app.HollandstormmodelSwitch.Tooltip = {''};
            app.HollandstormmodelSwitch.Position = [243 229 32 14];

            % Create WindwaveinteractionSwitchLabel
            app.WindwaveinteractionSwitchLabel = uilabel(app.Tab_4);
            app.WindwaveinteractionSwitchLabel.HorizontalAlignment = 'center';
            app.WindwaveinteractionSwitchLabel.Enable = 'off';
            app.WindwaveinteractionSwitchLabel.Tooltip = {'Activate wind-wave↙
interaction by Chen et al. (2003)'};
            app.WindwaveinteractionSwitchLabel.Position = [81 264 125 22];
            app.WindwaveinteractionSwitchLabel.Text = 'Wind-wave interaction';

            % Create WindwaveinteractionSwitch
            app.WindwaveinteractionSwitch = uiswitch(app.Tab_4, 'slider');
            app.WindwaveinteractionSwitch.ValueChangedFcn = createCallbackFcn(app, @WindwaveinteractionSwitchValueChanged, true);
            app.WindwaveinteractionSwitch.Enable = 'off';
            app.WindwaveinteractionSwitch.Tooltip = {''};
            app.WindwaveinteractionSwitch.Position = [243 268 32 14];
            app.WindwaveinteractionSwitch.Value = 'On';

            % Create StormfileEditField
            app.StormfileEditField = uieditfield(app.Tab_4, 'text');
            app.StormfileEditField.Editable = 'off';
            app.StormfileEditField.Enable = 'off';
            app.StormfileEditField.Tooltip = {'Import the storm file'};
            app.StormfileEditField.Position = [427 226 141 20];

            % Create StormfileEditFieldLabel
            app.StormfileEditFieldLabel = uilabel(app.Tab_4);
            app.StormfileEditFieldLabel.HorizontalAlignment = 'right';
            app.StormfileEditFieldLabel.Enable = 'off';
            app.StormfileEditFieldLabel.Tooltip = {'It must contain the variables used in the Holland model'};
            app.StormfileEditFieldLabel.Position = [341 225 57 22];
            app.StormfileEditFieldLabel.Text = 'Storm file';

            % Create PressurefileEditFieldLabel
            app.PressurefileEditFieldLabel = uilabel(app.Tab_4);
            app.PressurefileEditFieldLabel.HorizontalAlignment = 'right';
            app.PressurefileEditFieldLabel.Position = [344 429 71 22];
            app.PressurefileEditFieldLabel.Text = 'Pressure file';

            % Create PressurefileEditField
            app.PressurefileEditField = uieditfield(app.Tab_4, 'text');
            app.PressurefileEditField.Position = [427 429 141 20];

            % Create Button_43
            app.Button_43 = uibutton(app.Tab_4, 'push');
            app.Button_43.ButtonPushedFcn = createCallbackFcn(app, @Button_43Pushed, true);
            app.Button_43.Position = [575 429 22 20];
            app.Button_43.Text = '...';

            % Create PressureLabel
            app.PressureLabel = uilabel(app.Tab_4);
            app.PressureLabel.FontWeight = 'bold';
            app.PressureLabel.Position = [50 454 56 22];
            app.PressureLabel.Text = 'Pressure';

            % Create WindLabel
            app.WindLabel = uilabel(app.Tab_4);
            app.WindLabel.FontWeight = 'bold';
            app.WindLabel.Position = [50 370 34 22];
            app.WindLabel.Text = 'Wind';

            % Create Tab_3
            app.Tab_3 = uitab(app.TabGroup2);
            app.Tab_3.AutoResizeChildren = 'off';

            % Create Label_6
            app.Label_6 = uilabel(app.Tab_3);
            app.Label_6.BackgroundColor = [0.651 0.651 0.651];
            app.Label_6.Position = [1 62 738 22];
            app.Label_6.Text = '';

            % Create Label_5
            app.Label_5 = uilabel(app.Tab_3);
            app.Label_5.BackgroundColor = [0.651 0.651 0.651];
            app.Label_5.Position = [1 155 739 22];
            app.Label_5.Text = '';

            % Create Label_12
            app.Label_12 = uilabel(app.Tab_3);
            app.Label_12.BackgroundColor = [0.651 0.651 0.651];
            app.Label_12.Position = [2 495 738 22];
            app.Label_12.Text = '';

            % Create SedimentLabel
            app.SedimentLabel = uilabel(app.Tab_3);
            app.SedimentLabel.FontSize = 14;
            app.SedimentLabel.FontWeight = 'bold';
            app.SedimentLabel.FontColor = [1 1 1];
            app.SedimentLabel.Position = [35 495 69 22];
            app.SedimentLabel.Text = 'Sediment';

            % Create ButtonGroup_19
            app.ButtonGroup_19 = uibuttongroup(app.Tab_3);
            app.ButtonGroup_19.AutoResizeChildren = 'off';
            app.ButtonGroup_19.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_19SelectionChanged, true);
            app.ButtonGroup_19.BorderType = 'none';
            app.ButtonGroup_19.Position = [306 459 168 30];

            % Create UpwindingButton
            app.UpwindingButton = uiradiobutton(app.ButtonGroup_19);
            app.UpwindingButton.Text = 'Upwinding';
            app.UpwindingButton.Position = [10 6 80 22];
            app.UpwindingButton.Value = true;

            % Create TVDButton
            app.TVDButton = uiradiobutton(app.ButtonGroup_19);
            app.TVDButton.Tooltip = {'Total Variation Diminishing '};
            app.TVDButton.Text = 'TVD';
            app.TVDButton.Position = [102 6 45 22];

            % Create NumericalschemeLabel
            app.NumericalschemeLabel = uilabel(app.Tab_3);
            app.NumericalschemeLabel.Tooltip = {'Scheme to use for solving the advection-diffusion equation'};
            app.NumericalschemeLabel.Position = [187 463 105 22];
            app.NumericalschemeLabel.Text = 'Numerical scheme';

            % Create MediangrainDMEditFieldLabel
            app.MediangrainDMEditFieldLabel = uilabel(app.Tab_3);
            app.MediangrainDMEditFieldLabel.Tooltip = {'D50 of the grain diameter'};
            app.MediangrainDMEditFieldLabel.Position = [187 428 156 22];
            app.MediangrainDMEditFieldLabel.Text = 'Median grain diameter (mm)';

            % Create MediangrainDMEditField
            app.MediangrainDMEditField = uieditfield(app.Tab_3, 'numeric');
            app.MediangrainDMEditField.Limits = [0 Inf];
            app.MediangrainDMEditField.ValueDisplayFormat = '%11.3f';
            app.MediangrainDMEditField.ValueChangedFcn = createCallbackFcn(app, @MediangrainDMEditFieldValueChanged, true);
            app.MediangrainDMEditField.FontColor = [0.651 0.651 0.651];
            app.MediangrainDMEditField.Tooltip = {''};
            app.MediangrainDMEditField.Position = [490 429 63 19];
            app.MediangrainDMEditField.Value = 0.5;

            % Create SedimentdensityEditFieldLabel
            app.SedimentdensityEditFieldLabel = uilabel(app.Tab_3);
            app.SedimentdensityEditFieldLabel.Position = [187 398 99 22];
            app.SedimentdensityEditFieldLabel.Text = 'Sediment density';

            % Create SedimentdensityEditField
            app.SedimentdensityEditField = uieditfield(app.Tab_3, 'numeric');
            app.SedimentdensityEditField.Limits = [0 Inf];
            app.SedimentdensityEditField.ValueDisplayFormat = '%11.3f';
            app.SedimentdensityEditField.ValueChangedFcn = createCallbackFcn(app, @SedimentdensityEditFieldValueChanged, true);
            app.SedimentdensityEditField.FontColor = [0.651 0.651 0.651];
            app.SedimentdensityEditField.Position = [490 400 63 19];
            app.SedimentdensityEditField.Value = 2.68;

            % Create SedimentporosityEditFieldLabel
            app.SedimentporosityEditFieldLabel = uilabel(app.Tab_3);
            app.SedimentporosityEditFieldLabel.Position = [187 369 103 22];
            app.SedimentporosityEditFieldLabel.Text = 'Sediment porosity';

            % Create SedimentporosityEditField
            app.SedimentporosityEditField = uieditfield(app.Tab_3, 'numeric');
            app.SedimentporosityEditField.Limits = [0 Inf];
            app.SedimentporosityEditField.ValueDisplayFormat = '%11.3f';
            app.SedimentporosityEditField.ValueChangedFcn = createCallbackFcn(app, @SedimentporosityEditFieldValueChanged, true);
            app.SedimentporosityEditField.FontColor = [0.651 0.651 0.651];
            app.SedimentporosityEditField.Position = [490 370 63 19];
            app.SedimentporosityEditField.Value = 0.47;

            % Create SettlingvelocitymsEditFieldLabel
            app.SettlingvelocitymsEditFieldLabel = uilabel(app.Tab_3);
            app.SettlingvelocitymsEditFieldLabel.Tooltip = {'Rate at which sediment particles settles'};
            app.SettlingvelocitymsEditFieldLabel.Position = [187 340 120 22];
            app.SettlingvelocitymsEditFieldLabel.Text = 'Settling velocity (m/s)';

            % Create SettlingvelocitymsEditField
            app.SettlingvelocitymsEditField = uieditfield(app.Tab_3, 'numeric');
            app.SettlingvelocitymsEditField.Limits = [0 Inf];
            app.SettlingvelocitymsEditField.ValueDisplayFormat = '%11.3f';
            app.SettlingvelocitymsEditField.ValueChangedFcn =
createCallbackFcn(app, @SettlingvelocitymsEditFieldValueChanged, true);
            app.SettlingvelocitymsEditField.FontColor = [0.651 0.651 0.651];
            app.SettlingvelocitymsEditField.Tooltip = {''};
            app.SettlingvelocitymsEditField.Position = [490 342 63 19];
            app.SettlingvelocitymsEditField.Value = 0.125;

            % Create CriticalShieldsEditFieldLabel
            app.CriticalShieldsEditFieldLabel = uilabel(app.Tab_3);
            app.CriticalShieldsEditFieldLabel.Tooltip = {'Value defining when sediment particles begin to move under the influence of fluid flow'};
            app.CriticalShieldsEditFieldLabel.Position = [187 311 241 22];
            app.CriticalShieldsEditFieldLabel.Text = 'Critical Shields parameter (suspended load)';

            % Create CriticalShieldsEditField
            app.CriticalShieldsEditField = uieditfield(app.Tab_3, 'numeric');
            app.CriticalShieldsEditField.Limits = [0 Inf];
            app.CriticalShieldsEditField.ValueDisplayFormat = '%11.3f';
            app.CriticalShieldsEditField.ValueChangedFcn = createCallbackFcn(app, @CriticalShieldsEditFieldValueChanged, true);
            app.CriticalShieldsEditField.FontColor = [0.651 0.651 0.651];
            app.CriticalShieldsEditField.Tooltip = {''};
            app.CriticalShieldsEditField.Position = [490 313 63 19];
            app.CriticalShieldsEditField.Value = 0.05;

            % Create ShieldsparameterbedloadEditFieldLabel
            app.ShieldsparameterbedloadEditFieldLabel = uilabel(app.Tab_3);
            app.ShieldsparameterbedloadEditFieldLabel.Tooltip = {'Default value is based on Meyer-Peter and Muller (1984)'};
            app.ShieldsparameterbedloadEditFieldLabel.Position = [187 282 156 22];
            app.ShieldsparameterbedloadEditFieldLabel.Text = 'Shields parameter (bedload)';

            % Create ShieldsparamBedloadEditField
            app.ShieldsparamBedloadEditField = uieditfield(app.Tab_3, 'numeric');
            app.ShieldsparamBedloadEditField.Limits = [0 Inf];
            app.ShieldsparamBedloadEditField.ValueDisplayFormat = '%11.3f';
            app.ShieldsparamBedloadEditField.ValueChangedFcn = createCallbackFcn(app, @ShieldsparamBedloadEditFieldValueChanged, true);
            app.ShieldsparamBedloadEditField.FontColor = [0.651 0.651 0.651];
            app.ShieldsparamBedloadEditField.Tooltip = {''};
            app.ShieldsparamBedloadEditField.Position = [490 284 63 19];
            app.ShieldsparamBedloadEditField.Value = 0.047;

            % Create RungeKuttaparameter1EditFieldLabel
            app.RungeKuttaparameter1EditFieldLabel = uilabel(app.Tab_3);
            app.RungeKuttaparameter1EditFieldLabel.Position = [187 253 142 22];
            app.RungeKuttaparameter1EditFieldLabel.Text = 'Runge-Kutta parameter 1';

            % Create RungeKuttaparameter1EditField
            app.RungeKuttaparameter1EditField = uieditfield(app.Tab_3, 'numeric');
            app.RungeKuttaparameter1EditField.Limits = [0 Inf];
            app.RungeKuttaparameter1EditField.ValueDisplayFormat = '%11.4f';
            app.RungeKuttaparameter1EditField.ValueChangedFcn = createCallbackFcn(app, @RungeKuttaparameter1EditFieldValueChanged, true);
            app.RungeKuttaparameter1EditField.FontColor = [0.651 0.651 0.651];
            app.RungeKuttaparameter1EditField.Position = [490 254 63 19];
            app.RungeKuttaparameter1EditField.Value = 0.3333;

            % Create RungeKuttaparameter2EditFieldLabel
            app.RungeKuttaparameter2EditFieldLabel = uilabel(app.Tab_3);
            app.RungeKuttaparameter2EditFieldLabel.Position = [187 224 142 22];
            app.RungeKuttaparameter2EditFieldLabel.Text = 'Runge-Kutta parameter 2';

            % Create RungeKuttaparameter2EditField
            app.RungeKuttaparameter2EditField = uieditfield(app.Tab_3, 'numeric');
            app.RungeKuttaparameter2EditField.Limits = [0 Inf];
            app.RungeKuttaparameter2EditField.ValueDisplayFormat = '%11.3f';
            app.RungeKuttaparameter2EditField.ValueChangedFcn = createCallbackFcn(app, @RungeKuttaparameter2EditFieldValueChanged, true);
            app.RungeKuttaparameter2EditField.FontColor = [0.651 0.651 0.651];
            app.RungeKuttaparameter2EditField.Position = [490 226 63 19];
            app.RungeKuttaparameter2EditField.Value = 1;

            % Create MinDepthSedPickUpEditFieldLabel
            app.MinDepthSedPickUpEditFieldLabel = uilabel(app.Tab_3);
            app.MinDepthSedPickUpEditFieldLabel.Position = [187 195 259 22];
            app.MinDepthSedPickUpEditFieldLabel.Text = 'Minimum depth for sediment pickup action (m)';

            % Create MinDepthSedPickUpEditField
            app.MinDepthSedPickUpEditField = uieditfield(app.Tab_3, 'numeric');
            app.MinDepthSedPickUpEditField.Limits = [0 Inf];
            app.MinDepthSedPickUpEditField.ValueDisplayFormat = '%11.3f';
            app.MinDepthSedPickUpEditField.ValueChangedFcn = createCallbackFcn(app, @MinDepthSedPickUpEditFieldValueChanged, true);
            app.MinDepthSedPickUpEditField.FontColor = [0.651 0.651 0.651];
            app.MinDepthSedPickUpEditField.Position = [490 197 63 19];
            app.MinDepthSedPickUpEditField.Value = 0.1;

            % Create ButtonGroup_20
            app.ButtonGroup_20 = uibuttongroup(app.Tab_3);
            app.ButtonGroup_20.AutoResizeChildren = 'off';
            app.ButtonGroup_20.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_20SelectionChanged, true);
            app.ButtonGroup_20.BorderType = 'none';
            app.ButtonGroup_20.Position = [266 120 241 30];

            % Create UpdatedepthButton
            app.UpdatedepthButton = uiradiobutton(app.ButtonGroup_20);
            app.UpdatedepthButton.Tooltip = {'Bathymetry is updated in the simulation'};
            app.UpdatedepthButton.Text = ' Update depth';
            app.UpdatedepthButton.Position = [6 6 99 22];

            % Create NobedchangeButton
            app.NobedchangeButton = uiradiobutton(app.ButtonGroup_20);
            app.NobedchangeButton.Tooltip = {'Bathymetry remains constant'};
            app.NobedchangeButton.Text = ' No bed change';
            app.NobedchangeButton.Position = [121 6 108 22];

            % Create Button_36
            app.Button_36 = uiradiobutton(app.ButtonGroup_20);
            app.Button_36.Visible = 'off';
            app.Button_36.Text = '';
            app.Button_36.Position = [232 6 25 22];
            app.Button_36.Value = true;

            % Create BedChangeLabel
            app.BedChangeLabel = uilabel(app.Tab_3);
            app.BedChangeLabel.Position = [187 125 76 22];
            app.BedChangeLabel.Text = 'Bed Change:';

            % Create EditFieldLabel
            app.EditFieldLabel = uilabel(app.Tab_3);
            app.EditFieldLabel.HorizontalAlignment = 'right';
            app.EditFieldLabel.Enable = 'off';
            app.EditFieldLabel.Visible = 'off';
            app.EditFieldLabel.Position = [282 -4 341 22];
            app.EditFieldLabel.Text = 'Time step ratio: Boussinesq model to morphological updating';

            % Create TimeStepBouss
            app.TimeStepBouss = uieditfield(app.Tab_3, 'numeric');
            app.TimeStepBouss.Limits = [0 Inf];
            app.TimeStepBouss.ValueDisplayFormat = '%11.2f';
            app.TimeStepBouss.ValueChangedFcn = createCallbackFcn(app, @TimeStepBoussValueChanged, true);
            app.TimeStepBouss.Editable = 'off';
            app.TimeStepBouss.FontColor = [0.651 0.651 0.651];
            app.TimeStepBouss.Enable = 'off';
            app.TimeStepBouss.Visible = 'off';
            app.TimeStepBouss.Position = [636 -2 63 19];
            app.TimeStepBouss.Value = 25;

            % Create AvalancheLabel
            app.AvalancheLabel = uilabel(app.Tab_3);
            app.AvalancheLabel.FontSize = 14;
            app.AvalancheLabel.FontWeight = 'bold';
            app.AvalancheLabel.FontColor = [1 1 1];
            app.AvalancheLabel.Position = [35 62 74 22];
            app.AvalancheLabel.Text = 'Avalanche';

            % Create MorphologicalChangeLabel_2
            app.MorphologicalChangeLabel_2 = uilabel(app.Tab_3);
            app.MorphologicalChangeLabel_2.FontSize = 14;
            app.MorphologicalChangeLabel_2.FontWeight = 'bold';
            app.MorphologicalChangeLabel_2.FontColor = [1 1 1];
            app.MorphologicalChangeLabel_2.Position = [35 155 157 22];
            app.MorphologicalChangeLabel_2.Text = 'Morphological Change';

            % Create ConsideravalancheLabel
            app.ConsideravalancheLabel = uilabel(app.Tab_3);
            app.ConsideravalancheLabel.Position = [57 30 118 22];
            app.ConsideravalancheLabel.Text = 'Consider avalanche?';

            % Create ButtonGroup_22
            app.ButtonGroup_22 = uibuttongroup(app.Tab_3);
            app.ButtonGroup_22.AutoResizeChildren = 'off';
            app.ButtonGroup_22.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_22SelectionChanged, true);
            app.ButtonGroup_22.BorderType = 'none';
            app.ButtonGroup_22.Position = [181 27 150 30];

            % Create YesButton_8
            app.YesButton_8 = uiradiobutton(app.ButtonGroup_22);
            app.YesButton_8.Text = 'Yes';
            app.YesButton_8.Position = [9 5 41 22];

            % Create NoButton_7
            app.NoButton_7 = uiradiobutton(app.ButtonGroup_22);
            app.NoButton_7.Text = 'No';
            app.NoButton_7.Position = [61 5 38 22];
            app.NoButton_7.Value = true;

            % Create Button_38
            app.Button_38 = uiradiobutton(app.ButtonGroup_22);
            app.Button_38.Visible = 'off';
            app.Button_38.Text = '';
            app.Button_38.Position = [103 2 25 22];

            % Create RungeKuttaparam2EditFieldLabel_2
            app.RungeKuttaparam2EditFieldLabel_2 = uilabel(app.Tab_3);
            app.RungeKuttaparam2EditFieldLabel_2.Enable = 'off';
            app.RungeKuttaparam2EditFieldLabel_2.Tooltip = {'Steepest angle which the sediments remain stable'; ''; 'Default value 0.7 = 35 deg'};
            app.RungeKuttaparam2EditFieldLabel_2.Position = [392 30 154 22];
            app.RungeKuttaparam2EditFieldLabel_2.Text = 'Tangent of the repose angle';

            % Create RungeKuttaparam2EditField_2
            app.RungeKuttaparam2EditField_2 = uieditfield(app.Tab_3, 'numeric');
            app.RungeKuttaparam2EditField_2.ValueDisplayFormat = '%11.2f';
            app.RungeKuttaparam2EditField_2.ValueChangedFcn = createCallbackFcn(app, @RungeKuttaparam2EditField_2ValueChanged, true);
            app.RungeKuttaparam2EditField_2.FontColor = [0.651 0.651 0.651];
            app.RungeKuttaparam2EditField_2.Enable = 'off';
            app.RungeKuttaparam2EditField_2.Tooltip = {''};
            app.RungeKuttaparam2EditField_2.Position = [558 32 46 19];
            app.RungeKuttaparam2EditField_2.Value = 0.7;

            % Create FileEditFieldLabel
            app.FileEditFieldLabel = uilabel(app.Tab_3);
            app.FileEditFieldLabel.HorizontalAlignment = 'right';
            app.FileEditFieldLabel.Visible = 'off';
            app.FileEditFieldLabel.Position = [558 102 27 22];
            app.FileEditFieldLabel.Text = ' File';

            % Create FileEditField
            app.FileEditField = uieditfield(app.Tab_3, 'text');
            app.FileEditField.ValueChangedFcn = createCallbackFcn(app, @FileEditFieldValueChanged, true);
            app.FileEditField.Editable = 'off';
            app.FileEditField.Visible = 'off';
            app.FileEditField.Tooltip = {'Import the hard bottom file'; ''; 'File format and grid dimension must be the same as the input bathymetry'};
            app.FileEditField.Position = [592 103 96 20];

            % Create Button_35
            app.Button_35 = uibutton(app.Tab_3, 'push');
            app.Button_35.ButtonPushedFcn = createCallbackFcn(app, @Button_35Pushed, true);
            app.Button_35.Visible = 'off';
            app.Button_35.Tooltip = {''};
            app.Button_35.Position = [694 103 22 20];
            app.Button_35.Text = '...';

            % Create BottomTypeLabel
            app.BottomTypeLabel = uilabel(app.Tab_3);
            app.BottomTypeLabel.Position = [187 100 77 22];
            app.BottomTypeLabel.Text = 'Bottom Type:';

            % Create ButtonGroup_21
            app.ButtonGroup_21 = uibuttongroup(app.Tab_3);
            app.ButtonGroup_21.AutoResizeChildren = 'off';
            app.ButtonGroup_21.SelectionChangedFcn = createCallbackFcn(app, @ButtonGroup_21SelectionChanged, true);
            app.ButtonGroup_21.ForegroundColor = [0.651 0.651 0.651];
            app.ButtonGroup_21.BorderType = 'none';
            app.ButtonGroup_21.Position = [266 96 274 30];

            % Create HardbednoerosionButton
            app.HardbednoerosionButton = uiradiobutton(app.ButtonGroup_21);
            app.HardbednoerosionButton.Text = ' Hard bed, no erosion';
            app.HardbednoerosionButton.Position = [121 5 138 22];

            % Create SedimentbedButton
            app.SedimentbedButton = uiradiobutton(app.ButtonGroup_21);
            app.SedimentbedButton.Tooltip = {'Allow bed erosion'};
            app.SedimentbedButton.Text = ' Sediment bed';
            app.SedimentbedButton.Position = [6 5 100 22];

            % Create Button_37
            app.Button_37 = uiradiobutton(app.ButtonGroup_21);
            app.Button_37.Visible = 'off';
            app.Button_37.Text = '';
            app.Button_37.Position = [261 4 25 22];
            app.Button_37.Value = true;

            % Create TimeStepSedToMorpEditFieldLabel
            app.TimeStepSedToMorpEditFieldLabel = uilabel(app.Tab_3);
            app.TimeStepSedToMorpEditFieldLabel.HorizontalAlignment = 'right';
            app.TimeStepSedToMorpEditFieldLabel.Enable = 'off';
            app.TimeStepSedToMorpEditFieldLabel.Visible = 'off';
            app.TimeStepSedToMorpEditFieldLabel.Position = [331 194 237 22];
            app.TimeStepSedToMorpEditFieldLabel.Text = 'Time step ratio: sediment to morphological';

            % Create TimeStepSedToMorphlEditField
            app.TimeStepSedToMorphlEditField = uieditfield(app.Tab_3,
'numeric');
            app.TimeStepSedToMorphlEditField.Limits = [0 Inf];
            app.TimeStepSedToMorphlEditField.ValueChangedFcn = ↙
createCallbackFcn(app, @TimeStepSedToMorphlEditFieldValueChanged, true);
            app.TimeStepSedToMorphlEditField.Editable = 'off';
            app.TimeStepSedToMorphlEditField.FontColor = [0.651 0.651 0.651];
            app.TimeStepSedToMorphlEditField.Enable = 'off';
            app.TimeStepSedToMorphlEditField.Visible = 'off';
            app.TimeStepSedToMorphlEditField.Position = [491 187 63 19];
            app.TimeStepSedToMorphlEditField.Value = 10;

            % Create AddrollereffectsCheckBox
            app.AddrollereffectsCheckBox = uicheckbox(app.Tab_3);
            app.AddrollereffectsCheckBox.Text = 'Add roller effects';
            app.AddrollereffectsCheckBox.Position = [490 463 113 22];

            % Create Tab_6
            app.Tab_6 = uitab(app.TabGroup2);
            app.Tab_6.AutoResizeChildren = 'off';

            % Create Label_13
            app.Label_13 = uilabel(app.Tab_6);
            app.Label_13.BackgroundColor = [0.651 0.651 0.651];
            app.Label_13.Position = [1 495 738 22];
            app.Label_13.Text = '';

            % Create ImportVesselFilesLabel
            app.ImportVesselFilesLabel = uilabel(app.Tab_6);
            app.ImportVesselFilesLabel.Tooltip = {''};
            app.ImportVesselFilesLabel.Position = [77 452 106 22];
            app.ImportVesselFilesLabel.Text = 'Import vessel files:';

            % Create ImportvesselfilesTextArea
            app.ImportvesselfilesTextArea = uitextarea(app.Tab_6);
            app.ImportvesselfilesTextArea.Editable = 'off';
            app.ImportvesselfilesTextArea.Position = [77 324 154 120];

            % Create Button_40
            app.Button_40 = uibutton(app.Tab_6, 'push');
            app.Button_40.ButtonPushedFcn = createCallbackFcn(app, ↙
@Button_40Pushed, true);
            app.Button_40.Tooltip = {'Find vessel files'};
            app.Button_40.Position = [240 374 26 20];
            app.Button_40.Text = '...';

            % Create VesselLabel
            app.VesselLabel = uilabel(app.Tab_6);
            app.VesselLabel.FontSize = 14;
            app.VesselLabel.FontWeight = 'bold';
            app.VesselLabel.FontColor = [1 1 1];
            app.VesselLabel.Position = [24 495 48 22];
            app.VesselLabel.Text = 'Vessel';

            % Create Label_8
            app.Label_8 = uilabel(app.Tab_6);
            app.Label_8.BackgroundColor = [0.651 0.651 0.651];
            app.Label_8.Position = [1 197 738 22];
            app.Label_8.Text = '';
```

```matlab

            % Create DeepDraftVesselsLabel
            app.DeepDraftVesselsLabel = uilabel(app.Tab_6);
            app.DeepDraftVesselsLabel.FontWeight = 'bold';
            app.DeepDraftVesselsLabel.FontColor = [1 1 1];
            app.DeepDraftVesselsLabel.Position = [24 197 114 22];
            app.DeepDraftVesselsLabel.Text = 'Deep Draft Vessels';

            % Create MinimumClearanceLabel
            app.MinimumClearanceLabel = uilabel(app.Tab_6);
            app.MinimumClearanceLabel.Enable = 'off';
            app.MinimumClearanceLabel.Position = [80 132 116 22];
            app.MinimumClearanceLabel.Text = 'Minimum Clearance:';

            % Create MinimumClearanceEditField
            app.MinimumClearanceEditField = uieditfield(app.Tab_6, 'numeric');
            app.MinimumClearanceEditField.Limits = [0 Inf];
            app.MinimumClearanceEditField.ValueDisplayFormat = '%5.1f';
            app.MinimumClearanceEditField.ValueChangedFcn = createCallbackFcn(app, @MinimumClearanceEditFieldValueChanged, true);
            app.MinimumClearanceEditField.FontColor = [0.651 0.651 0.651];
            app.MinimumClearanceEditField.Enable = 'off';
            app.MinimumClearanceEditField.Position = [198 134 35 19];
            app.MinimumClearanceEditField.Value = 1;

            % Create IncludeSedimentEffectintheModelCheckBox
            app.IncludeSedimentEffectintheModelCheckBox = uicheckbox(app.Tab_6);
            app.IncludeSedimentEffectintheModelCheckBox.Text = 'Include Sediment Effect in the Model';
            app.IncludeSedimentEffectintheModelCheckBox.Position = [75 296 221 22];

            % Create ActivateCheckBox
            app.ActivateCheckBox = uicheckbox(app.Tab_6);
            app.ActivateCheckBox.ValueChangedFcn = createCallbackFcn(app, @ActivateCheckBoxValueChanged, true);
            app.ActivateCheckBox.Tooltip = {'Check the box to address the instability issue that may arise when modelling a large vessel with a draft close to the channel depth'};
            app.ActivateCheckBox.Text = '  Activate to reduce high-frequency spikes occurring close to the ship';
            app.ActivateCheckBox.Position = [81 163 401 22];

            % Create ShockcapturingCheckBox
            app.ShockcapturingCheckBox = uicheckbox(app.Tab_6);
            app.ShockcapturingCheckBox.Enable = 'off';
            app.ShockcapturingCheckBox.Tooltip = {'Utilizes shock-capturing method'};
            app.ShockcapturingCheckBox.Text = 'Shock capturing';
            app.ShockcapturingCheckBox.Position = [451 72 110 22];
            app.ShockcapturingCheckBox.Value = true;

            % Create FrictionCheckBox_2
            app.FrictionCheckBox_2 = uicheckbox(app.Tab_6);
            app.FrictionCheckBox_2.ValueChangedFcn = createCallbackFcn(app, @FrictionCheckBox_2ValueChanged, true);
            app.FrictionCheckBox_2.Enable = 'off';
            app.FrictionCheckBox_2.Tooltip = {'Utilizes dissipation method for damping waves similar to a sponge layer'};
            app.FrictionCheckBox_2.Text = 'Friction';
            app.FrictionCheckBox_2.Position = [80 72 62 22];
            app.FrictionCheckBox_2.Value = true;

            % Create ViscosityCheckBox
            app.ViscosityCheckBox = uicheckbox(app.Tab_6);
            app.ViscosityCheckBox.ValueChangedFcn = createCallbackFcn(app, @ViscosityCheckBoxValueChanged, true);
            app.ViscosityCheckBox.Enable = 'off';
            app.ViscosityCheckBox.Tooltip = {'Similar to friction method but its wave damping rate is usually lower'};
            app.ViscosityCheckBox.Text = 'Viscosity';
            app.ViscosityCheckBox.Position = [279 72 70 22];

            % Create FrictionCoefficientEditFieldLabel
            app.FrictionCoefficientEditFieldLabel = uilabel(app.Tab_6);
            app.FrictionCoefficientEditFieldLabel.HorizontalAlignment = 'right';
            app.FrictionCoefficientEditFieldLabel.Enable = 'off';
            app.FrictionCoefficientEditFieldLabel.Tooltip = {'Suggested values: 0.1 - 1'};
            app.FrictionCoefficientEditFieldLabel.Position = [136 74 63 19];
            app.FrictionCoefficientEditFieldLabel.Text = 'Coefficient';

            % Create FrictionCoefficientEditField
            app.FrictionCoefficientEditField = uieditfield(app.Tab_6, 'numeric');
            app.FrictionCoefficientEditField.Limits = [0 Inf];
            app.FrictionCoefficientEditField.ValueDisplayFormat = '%5.1f';
            app.FrictionCoefficientEditField.ValueChangedFcn = createCallbackFcn(app, @FrictionCoefficientEditFieldValueChanged, true);
            app.FrictionCoefficientEditField.FontColor = [0.651 0.651 0.651];
            app.FrictionCoefficientEditField.Enable = 'off';
            app.FrictionCoefficientEditField.Tooltip = {''};
            app.FrictionCoefficientEditField.Position = [209 74 35 19];
            app.FrictionCoefficientEditField.Value = 0.1;

            % Create ValueEditFieldLabel_2
            app.ValueEditFieldLabel_2 = uilabel(app.Tab_6);
            app.ValueEditFieldLabel_2.HorizontalAlignment = 'right';
            app.ValueEditFieldLabel_2.Enable = 'off';
            app.ValueEditFieldLabel_2.Tooltip = {'Suggested values: 0.1 - 5'};
            app.ValueEditFieldLabel_2.Position = [343 74 34 19];
            app.ValueEditFieldLabel_2.Text = 'Value';

            % Create ValueEditField
            app.ValueEditField = uieditfield(app.Tab_6, 'numeric');
            app.ValueEditField.Limits = [0 Inf];
            app.ValueEditField.ValueDisplayFormat = '%5.1f';
            app.ValueEditField.ValueChangedFcn = createCallbackFcn(app, @ValueEditFieldValueChanged, true);
            app.ValueEditField.Editable = 'off';
            app.ValueEditField.FontColor = [0.651 0.651 0.651];
            app.ValueEditField.Enable = 'off';
            app.ValueEditField.Tooltip = {''};
            app.ValueEditField.Position = [384 74 35 19];
            app.ValueEditField.Value = 1;

            % Create SelectMethodLabel
            app.SelectMethodLabel = uilabel(app.Tab_6);
            app.SelectMethodLabel.FontWeight = 'bold';
            app.SelectMethodLabel.Enable = 'off';
            app.SelectMethodLabel.Position = [81 97 92 22];
            app.SelectMethodLabel.Text = 'Select Method:';

            % Create SelectInitialConditonLabel
            app.SelectInitialConditonLabel = uilabel(app.InitialConditionTab);
            app.SelectInitialConditonLabel.FontWeight = 'bold';
            app.SelectInitialConditonLabel.Position = [21 614 135 22];
            app.SelectInitialConditonLabel.Text = 'Select Initial Conditon:';

            % Create ButtonGroup_18
            app.ButtonGroup_18 = uibuttongroup(app.InitialConditionTab);
            app.ButtonGroup_18.AutoResizeChildren = 'off';
            app.ButtonGroup_18.SelectionChangedFcn = createCallbackFcn(app, ↙
@ButtonGroup_6SelectionChanged, true);
            app.ButtonGroup_18.BorderType = 'none';
            app.ButtonGroup_18.FontSize = 14;
            app.ButtonGroup_18.Position = [167 610 412 30];

            % Create ImportUVZButton
            app.ImportUVZButton = uiradiobutton(app.ButtonGroup_18);
            app.ImportUVZButton.Text = 'Import UVZ';
            app.ImportUVZButton.Position = [3 5 84 22];

            % Create WavemakerButton
            app.WavemakerButton = uiradiobutton(app.ButtonGroup_18);
            app.WavemakerButton.Text = 'Wavemaker';
            app.WavemakerButton.Position = [109 5 85 22];

            % Create Button_31
            app.Button_31 = uiradiobutton(app.ButtonGroup_18);
            app.Button_31.Enable = 'off';
            app.Button_31.Visible = 'off';
            app.Button_31.Text = '';
            app.Button_31.Position = [395 5 25 22];
            app.Button_31.Value = true;

            % Create MeteotsunamiButton
            app.MeteotsunamiButton = uiradiobutton(app.ButtonGroup_18);
            app.MeteotsunamiButton.Tooltip = {'Initiate the tsunami using↙
atmospheric disturbances'};
            app.MeteotsunamiButton.Text = 'Meteotsunami';
            app.MeteotsunamiButton.Position = [215 5 98 22];

            % Create VesselButton
            app.VesselButton = uiradiobutton(app.ButtonGroup_18);
            app.VesselButton.Tooltip = {'Use the ship-wake model in the↙
simulation'};
            app.VesselButton.Text = 'Vessel';
            app.VesselButton.Position = [334 5 56 22];

            % Create GenerateinputtxtButton_4
            app.GenerateinputtxtButton_4 = uibutton(app.InitialConditionTab, ↙
'push');
            app.GenerateinputtxtButton_4.ButtonPushedFcn = createCallbackFcn↙
(app, @GenerateinputtxtButton_4Pushed, true);
            app.GenerateinputtxtButton_4.HandleVisibility = 'off';
            app.GenerateinputtxtButton_4.BackgroundColor = [0.9216 0.9216↙
0.9216];
            app.GenerateinputtxtButton_4.FontSize = 15;
            app.GenerateinputtxtButton_4.FontWeight = 'bold';
            app.GenerateinputtxtButton_4.Position = [186 23 355 43];
            app.GenerateinputtxtButton_4.Text = 'Generate input.txt';

            % Create SedimentTransportCheckBox
            app.SedimentTransportCheckBox = uicheckbox(app.↙
InitialConditionTab);
            app.SedimentTransportCheckBox.ValueChangedFcn = createCallbackFcn↙
(app, @SedimentTransportCheckBoxValueChanged, true);
            app.SedimentTransportCheckBox.Text = 'Sediment Transport';
            app.SedimentTransportCheckBox.Position = [584 614 127 22];

            % Create ExpectedOutputTab
            app.ExpectedOutputTab = uitab(app.TabGroup);
            app.ExpectedOutputTab.AutoResizeChildren = 'off';
            app.ExpectedOutputTab.Tooltip = {'Flux F in X direction'};
            app.ExpectedOutputTab.Title = 'Expected Output';
            app.ExpectedOutputTab.BackgroundColor = [0.9412 0.9412 0.9412];
            app.ExpectedOutputTab.Scrollable = 'on';

            % Create GridLayout
            app.GridLayout = uigridlayout(app.ExpectedOutputTab);
            app.GridLayout.ColumnWidth = {'1x', 31, 45, 36, 39, 38, 54, 67,↙
66, 76, 57, '2.84x'};
            app.GridLayout.RowHeight = {22, 22, 22, 22, 22, 22, '1x', 22, 22,↙
22, 22, 22, 22, 22, 22, 22, '1.75x', 43};

            % Create DepthCheckBox
            app.DepthCheckBox = uicheckbox(app.GridLayout);
            app.DepthCheckBox.ValueChangedFcn = createCallbackFcn(app,↙
@DepthCheckBoxValueChanged, true);
            app.DepthCheckBox.Tooltip = {'DEPTH_OUT'};
            app.DepthCheckBox.Text = 'Depth';
            app.DepthCheckBox.Layout.Row = 6;
            app.DepthCheckBox.Layout.Column = 7;

            % Create MaxVorticityCheckBox
            app.MaxVorticityCheckBox = uicheckbox(app.GridLayout);
            app.MaxVorticityCheckBox.ValueChangedFcn = createCallbackFcn(app,↙
@MaxVorticityCheckBoxValueChanged, true);
            app.MaxVorticityCheckBox.Tooltip = {'VORMAX'};
            app.MaxVorticityCheckBox.Text = 'Maximum Vorticity';
            app.MaxVorticityCheckBox.Layout.Row = 9;
            app.MaxVorticityCheckBox.Layout.Column = [7 8];

            % Create UVelocityCheckBox
            app.UVelocityCheckBox = uicheckbox(app.GridLayout);
            app.UVelocityCheckBox.ValueChangedFcn = createCallbackFcn(app,↙
@UVelocityCheckBoxValueChanged, true);
            app.UVelocityCheckBox.Tooltip = {'U: Velocity in X direction'};
            app.UVelocityCheckBox.Text = 'U Velocity';
            app.UVelocityCheckBox.Layout.Row = 2;
            app.UVelocityCheckBox.Layout.Column = 10;

            % Create MaxMomentumFluxCheckBox
            app.MaxMomentumFluxCheckBox = uicheckbox(app.GridLayout);
            app.MaxMomentumFluxCheckBox.ValueChangedFcn = createCallbackFcn(app, @MaxMomentumFluxCheckBoxValueChanged, true);
            app.MaxMomentumFluxCheckBox.Tooltip = {'MFMAX'};
            app.MaxMomentumFluxCheckBox.Text = 'Maximum Momentum Flux';
            app.MaxMomentumFluxCheckBox.Layout.Row = 11;
            app.MaxMomentumFluxCheckBox.Layout.Column = [2 5];

            % Create VVelocityCheckBox
            app.VVelocityCheckBox = uicheckbox(app.GridLayout);
            app.VVelocityCheckBox.ValueChangedFcn = createCallbackFcn(app, @VVelocityCheckBoxValueChanged, true);
            app.VVelocityCheckBox.Tooltip = {' in y direction'};
            app.VVelocityCheckBox.Text = 'V Velocity';
            app.VVelocityCheckBox.Layout.Row = 3;
            app.VVelocityCheckBox.Layout.Column = 10;

            % Create ArrivalCheckBox
            app.ArrivalCheckBox = uicheckbox(app.GridLayout);
            app.ArrivalCheckBox.ValueChangedFcn = createCallbackFcn(app, @ArrivalCheckBoxValueChanged, true);
            app.ArrivalCheckBox.Tooltip = {'OUT_Time'};
            app.ArrivalCheckBox.Text = 'Arrival';
            app.ArrivalCheckBox.Layout.Row = 2;
            app.ArrivalCheckBox.Layout.Column = [7 8];

            % Create SurfaceElevationattimetCheckBox
            app.SurfaceElevationattimetCheckBox = uicheckbox(app.GridLayout);
            app.SurfaceElevationattimetCheckBox.ValueChangedFcn = createCallbackFcn(app, @SurfaceElevationattimetCheckBoxValueChanged, true);
            app.SurfaceElevationattimetCheckBox.Tooltip = {'ETA: Surface elevation at time t'};
            app.SurfaceElevationattimetCheckBox.Text = 'Surface Elevation at time t';
            app.SurfaceElevationattimetCheckBox.Layout.Row = 4;
            app.SurfaceElevationattimetCheckBox.Layout.Column = [2 5];
            app.SurfaceElevationattimetCheckBox.Value = true;

            % Create WetdrymaskCheckBox
            app.WetdrymaskCheckBox = uicheckbox(app.GridLayout);
            app.WetdrymaskCheckBox.ValueChangedFcn = createCallbackFcn(app, @WetdrymaskCheckBoxValueChanged, true);
            app.WetdrymaskCheckBox.Tooltip = {'MASK'};
            app.WetdrymaskCheckBox.Text = 'Wet-dry mask';
            app.WetdrymaskCheckBox.Layout.Row = 9;
            app.WetdrymaskCheckBox.Layout.Column = [10 11];

            % Create WaveHeightCheckBox
            app.WaveHeightCheckBox = uicheckbox(app.GridLayout);
            app.WaveHeightCheckBox.ValueChangedFcn = createCallbackFcn(app, @WaveHeightCheckBoxValueChanged, true);
            app.WaveHeightCheckBox.Tooltip = {'Wave height: Hsig,Hrms, Havg'};
            app.WaveHeightCheckBox.Text = 'Wave Height';
            app.WaveHeightCheckBox.Layout.Row = 5;
            app.WaveHeightCheckBox.Layout.Column = [2 3];

            % Create WetdrymaskforBoussinesqNSWECheckBox
            app.WetdrymaskforBoussinesqNSWECheckBox = uicheckbox(app.GridLayout);
            app.WetdrymaskforBoussinesqNSWECheckBox.ValueChangedFcn = createCallbackFcn(app, @WetdrymaskforBoussinesqNSWECheckBoxValueChanged, true);
            app.WetdrymaskforBoussinesqNSWECheckBox.Tooltip = {'MASK9: Crate a wetting-drying mask for Boussinesq/NSWE modelling'};
            app.WetdrymaskforBoussinesqNSWECheckBox.Text = 'Wet-dry mask for Boussinesq/NSWE';
            app.WetdrymaskforBoussinesqNSWECheckBox.Layout.Row = 10;
            app.WetdrymaskforBoussinesqNSWECheckBox.Layout.Column = [10 12];

            % Create RollerinducedFluxCheckBox
            app.RollerinducedFluxCheckBox = uicheckbox(app.GridLayout);
            app.RollerinducedFluxCheckBox.ValueChangedFcn = createCallbackFcn(app, @RollerinducedFluxCheckBoxValueChanged, true);
            app.RollerinducedFluxCheckBox.Tooltip = {'ROLLER: Roller-induced mass flux'};
            app.RollerinducedFluxCheckBox.Text = 'Roller-induced Flux';
            app.RollerinducedFluxCheckBox.Layout.Row = 16;
            app.RollerinducedFluxCheckBox.Layout.Column = [2 4];

            % Create XSourceCheckBox
            app.XSourceCheckBox = uicheckbox(app.GridLayout);
            app.XSourceCheckBox.ValueChangedFcn = createCallbackFcn(app, @XSourceCheckBoxValueChanged, true);
            app.XSourceCheckBox.Tooltip = {'SoureceX: Source terms in X direction'};
            app.XSourceCheckBox.Text = 'X Source';
            app.XSourceCheckBox.Layout.Row = 4;
            app.XSourceCheckBox.Layout.Column = [7 8];

            % Create PressureFieldCheckBox
            app.PressureFieldCheckBox = uicheckbox(app.GridLayout);
            app.PressureFieldCheckBox.ValueChangedFcn = createCallbackFcn(app, @PressureFieldCheckBoxValueChanged, true);
            app.PressureFieldCheckBox.Tooltip = {'OUT_METEO'};
            app.PressureFieldCheckBox.Text = 'Pressure Field';
            app.PressureFieldCheckBox.Layout.Row = 13;
            app.PressureFieldCheckBox.Layout.Column = [7 8];

            % Create YSourceCheckBox
            app.YSourceCheckBox = uicheckbox(app.GridLayout);
            app.YSourceCheckBox.ValueChangedFcn = createCallbackFcn(app, @YSourceCheckBoxValueChanged, true);
            app.YSourceCheckBox.Tooltip = {'SoureceY: Source terms in Y direction'};
            app.YSourceCheckBox.Text = 'Y Source';
            app.YSourceCheckBox.Layout.Row = 5;
            app.YSourceCheckBox.Layout.Column = [7 8];

            % Create BreakingLocationCheckBox
            app.BreakingLocationCheckBox = uicheckbox(app.GridLayout);
            app.BreakingLocationCheckBox.ValueChangedFcn = createCallbackFcn(app, @BreakingLocationCheckBoxValueChanged, true);
            app.BreakingLocationCheckBox.Tooltip = {'OUT_NU'};
            app.BreakingLocationCheckBox.Text = 'Breaking Location';
            app.BreakingLocationCheckBox.Layout.Row = 11;
            app.BreakingLocationCheckBox.Layout.Column = [7 8];

            % Create PMomentumFluxCheckBox
            app.PMomentumFluxCheckBox = uicheckbox(app.GridLayout);
            app.PMomentumFluxCheckBox.ValueChangedFcn = createCallbackFcn(app, @PMomentumFluxCheckBoxValueChanged, true);
            app.PMomentumFluxCheckBox.Tooltip = {'P: Momentum flux in X direction'};
            app.PMomentumFluxCheckBox.Text = 'P Momentum Flux';
            app.PMomentumFluxCheckBox.Layout.Row = 9;
            app.PMomentumFluxCheckBox.Layout.Column = [2 4];

            % Create QmomentumFluxCheckBox
            app.QmomentumFluxCheckBox = uicheckbox(app.GridLayout);
            app.QmomentumFluxCheckBox.ValueChangedFcn = createCallbackFcn(app, @QmomentumFluxCheckBoxValueChanged, true);
            app.QmomentumFluxCheckBox.Tooltip = {'Q: Momentum flux in Y direction'};
            app.QmomentumFluxCheckBox.Text = 'Q momentum Flux';
            app.QmomentumFluxCheckBox.Layout.Row = 10;
            app.QmomentumFluxCheckBox.Layout.Column = [2 4];

            % Create FxFluxCheckBox
            app.FxFluxCheckBox = uicheckbox(app.GridLayout);
            app.FxFluxCheckBox.ValueChangedFcn = createCallbackFcn(app, @FxFluxCheckBoxValueChanged, true);
            app.FxFluxCheckBox.Tooltip = {'Flux F in X direction'};
            app.FxFluxCheckBox.Text = 'Fx Flux';
            app.FxFluxCheckBox.Layout.Row = 12;
            app.FxFluxCheckBox.Layout.Column = [2 3];

            % Create FyFluxCheckBox
            app.FyFluxCheckBox = uicheckbox(app.GridLayout);
            app.FyFluxCheckBox.ValueChangedFcn = createCallbackFcn(app, @FyFluxCheckBoxValueChanged, true);
            app.FyFluxCheckBox.Tooltip = {'Flux F in Y direction'};
            app.FyFluxCheckBox.Text = 'Fy Flux';
            app.FyFluxCheckBox.Layout.Row = 13;
            app.FyFluxCheckBox.Layout.Column = [2 3];

            % Create GxFluxCheckBox
            app.GxFluxCheckBox = uicheckbox(app.GridLayout);
            app.GxFluxCheckBox.ValueChangedFcn = createCallbackFcn(app, @GxFluxCheckBoxValueChanged, true);
            app.GxFluxCheckBox.Tooltip = {'Gx: Flux G in X direction'};
            app.GxFluxCheckBox.Text = 'Gx Flux';
            app.GxFluxCheckBox.Layout.Row = 14;
            app.GxFluxCheckBox.Layout.Column = [2 3];

            % Create GyFluxCheckBox
            app.GyFluxCheckBox = uicheckbox(app.GridLayout);
            app.GyFluxCheckBox.ValueChangedFcn = createCallbackFcn(app, @GyFluxCheckBoxValueChanged, true);
            app.GyFluxCheckBox.Tooltip = {'Gy: Flux G in Y direction'};
            app.GyFluxCheckBox.Text = 'Gy Flux';
            app.GyFluxCheckBox.Layout.Row = 15;
            app.GyFluxCheckBox.Layout.Column = [2 3];

            % Create BreakingAgeCheckBox
            app.BreakingAgeCheckBox = uicheckbox(app.GridLayout);
            app.BreakingAgeCheckBox.ValueChangedFcn = createCallbackFcn(app, ↙
@BreakingAgeCheckBoxValueChanged, true);
            app.BreakingAgeCheckBox.Tooltip = {'AGE: Breaking Age'};
            app.BreakingAgeCheckBox.Text = 'Breaking Age';
            app.BreakingAgeCheckBox.Layout.Row = 10;
            app.BreakingAgeCheckBox.Layout.Column = [7 8];

            % Create MaximumVelocityCheckBox
            app.MaximumVelocityCheckBox = uicheckbox(app.GridLayout);
            app.MaximumVelocityCheckBox.ValueChangedFcn = createCallbackFcn↙
(app, @MaximumVelocityCheckBoxValueChanged, true);
            app.MaximumVelocityCheckBox.Tooltip = {'Umax: Maximum velocity'};
            app.MaximumVelocityCheckBox.Text = 'Maximum Velocity ';
            app.MaximumVelocityCheckBox.Layout.Row = 6;
            app.MaximumVelocityCheckBox.Layout.Column = [10 11];

            % Create UndertowCheckBox
            app.UndertowCheckBox = uicheckbox(app.GridLayout);
            app.UndertowCheckBox.ValueChangedFcn = createCallbackFcn(app, ↙
@UndertowCheckBoxValueChanged, true);
            app.UndertowCheckBox.Tooltip = {'UNDERTOW: Roller-induced extra↙
undertow flux'};
            app.UndertowCheckBox.Text = 'Undertow';
            app.UndertowCheckBox.Layout.Row = 12;
            app.UndertowCheckBox.Layout.Column = [7 8];

            % Create VelocityLabel
            app.VelocityLabel = uilabel(app.GridLayout);
            app.VelocityLabel.FontSize = 13;
            app.VelocityLabel.FontWeight = 'bold';
            app.VelocityLabel.FontColor = [0.0314 0.3686 0.6];
            app.VelocityLabel.Layout.Row = 1;
            app.VelocityLabel.Layout.Column = 10;
            app.VelocityLabel.Text = 'Velocity';

            % Create WaveHeightLabel
            app.WaveHeightLabel = uilabel(app.GridLayout);
            app.WaveHeightLabel.FontSize = 13;
            app.WaveHeightLabel.FontWeight = 'bold';
            app.WaveHeightLabel.FontColor = [0.0314 0.3686 0.6];
            app.WaveHeightLabel.Layout.Row = 1;
            app.WaveHeightLabel.Layout.Column = [2 3];
            app.WaveHeightLabel.Text = 'Wave Height';

            % Create SourceLabel
            app.SourceLabel = uilabel(app.GridLayout);
            app.SourceLabel.FontSize = 13;
            app.SourceLabel.FontWeight = 'bold';
            app.SourceLabel.FontColor = [0.0314 0.3686 0.6];
            app.SourceLabel.Layout.Row = 3;
            app.SourceLabel.Layout.Column = 7;
            app.SourceLabel.Text = 'Source';

            % Create MaskLabel
            app.MaskLabel = uilabel(app.GridLayout);
            app.MaskLabel.FontSize = 13;
            app.MaskLabel.FontWeight = 'bold';
            app.MaskLabel.FontColor = [0.0314 0.3686 0.6];
            app.MaskLabel.Layout.Row = 8;
            app.MaskLabel.Layout.Column = 10;
            app.MaskLabel.Text = 'Mask';

            % Create FluxLabel
            app.FluxLabel = uilabel(app.GridLayout);
            app.FluxLabel.FontSize = 13;
            app.FluxLabel.FontWeight = 'bold';
            app.FluxLabel.FontColor = [0.0314 0.3686 0.6];
            app.FluxLabel.Layout.Row = 8;
            app.FluxLabel.Layout.Column = 2;
            app.FluxLabel.Text = 'Flux';

            % Create TimeLabel
            app.TimeLabel = uilabel(app.GridLayout);
            app.TimeLabel.FontSize = 13;
            app.TimeLabel.FontWeight = 'bold';
            app.TimeLabel.FontColor = [0.0314 0.3686 0.6];
            app.TimeLabel.Layout.Row = 1;
            app.TimeLabel.Layout.Column = 7;
            app.TimeLabel.Text = 'Time';

            % Create OthersLabel
            app.OthersLabel = uilabel(app.GridLayout);
            app.OthersLabel.FontSize = 13;
            app.OthersLabel.FontWeight = 'bold';
            app.OthersLabel.FontColor = [0.0314 0.3686 0.6];
            app.OthersLabel.Layout.Row = 8;
            app.OthersLabel.Layout.Column = 7;
            app.OthersLabel.Text = 'Others';

            % Create GenerateinputtxtButton_2
            app.GenerateinputtxtButton_2 = uibutton(app.GridLayout, 'push');
            app.GenerateinputtxtButton_2.ButtonPushedFcn = createCallbackFcn(app, @GenerateinputtxtButton_2Pushed, true);
            app.GenerateinputtxtButton_2.HandleVisibility = 'off';
            app.GenerateinputtxtButton_2.BackgroundColor = [0.9216 0.9216 0.9216];
            app.GenerateinputtxtButton_2.FontSize = 15;
            app.GenerateinputtxtButton_2.FontWeight = 'bold';
            app.GenerateinputtxtButton_2.Layout.Row = 18;
            app.GenerateinputtxtButton_2.Layout.Column = [4 10];
            app.GenerateinputtxtButton_2.Text = 'Generate input.txt';

            % Create UVelocitymeanCheckBox
            app.UVelocitymeanCheckBox = uicheckbox(app.GridLayout);
            app.UVelocitymeanCheckBox.Tooltip = {'Umean: Mean velocity in X direction'};
            app.UVelocitymeanCheckBox.Text = 'U Velocity (mean)';
            app.UVelocitymeanCheckBox.Layout.Row = 4;
            app.UVelocitymeanCheckBox.Layout.Column = [10 11];

            % Create VVelocitymeanCheckBox
            app.VVelocitymeanCheckBox = uicheckbox(app.GridLayout);
            app.VVelocitymeanCheckBox.Tooltip = {'Vmean: Mean velocity in Y direction'};
            app.VVelocitymeanCheckBox.Text = 'V Velocity (mean)';
            app.VVelocitymeanCheckBox.Layout.Row = 5;
            app.VVelocitymeanCheckBox.Layout.Column = [10 11];

            % Create MinimumWaveHeightCheckBox
            app.MinimumWaveHeightCheckBox = uicheckbox(app.GridLayout);
            app.MinimumWaveHeightCheckBox.ValueChangedFcn = createCallbackFcn(app, @MinimumWaveHeightCheckBoxValueChanged, true);
            app.MinimumWaveHeightCheckBox.Tooltip = {'Hmin: Minimum height within the total simulation time'};
            app.MinimumWaveHeightCheckBox.Text = 'Minimum Wave Height';
            app.MinimumWaveHeightCheckBox.Layout.Row = 3;
            app.MinimumWaveHeightCheckBox.Layout.Column = [2 5];

            % Create MaximumWaveHeightCheckBox
            app.MaximumWaveHeightCheckBox = uicheckbox(app.GridLayout);
            app.MaximumWaveHeightCheckBox.ValueChangedFcn = createCallbackFcn(app, @MaximumWaveHeightCheckBoxValueChanged, true);
            app.MaximumWaveHeightCheckBox.Tooltip = {'Hmax: Maximum height within the total simulation time'};
            app.MaximumWaveHeightCheckBox.Text = 'Maximum Wave Height';
            app.MaximumWaveHeightCheckBox.Layout.Row = 2;
            app.MaximumWaveHeightCheckBox.Layout.Column = [2 5];
            app.MaximumWaveHeightCheckBox.Value = true;

            % Create ThresholdEditFieldLabel
            app.ThresholdEditFieldLabel = uilabel(app.GridLayout);
            app.ThresholdEditFieldLabel.VerticalAlignment = 'bottom';
            app.ThresholdEditFieldLabel.Visible = 'off';
            app.ThresholdEditFieldLabel.Layout.Row = 1;
            app.ThresholdEditFieldLabel.Layout.Column = 8;
            app.ThresholdEditFieldLabel.Text = 'Threshold';

            % Create ThresholdEditField
            app.ThresholdEditField = uieditfield(app.GridLayout, 'numeric');
            app.ThresholdEditField.Limits = [0.0001 Inf];
            app.ThresholdEditField.ValueChangedFcn = createCallbackFcn(app, @ThresholdEditFieldValueChanged, true);
            app.ThresholdEditField.FontColor = [0.651 0.651 0.651];
            app.ThresholdEditField.Visible = 'off';
            app.ThresholdEditField.Tooltip = {'Threshold to record the arrival time of the tsunami wave'; ''; 'Unit: meters'};
            app.ThresholdEditField.Layout.Row = 2;
            app.ThresholdEditField.Layout.Column = 8;
            app.ThresholdEditField.Value = 0.0001;

            % Create Panel_32
            app.Panel_32 = uipanel(app.UIFigure);
            app.Panel_32.AutoResizeChildren = 'off';
            app.Panel_32.SizeChangedFcn = createCallbackFcn(app, @Panel_32SizeChanged, true);
            app.Panel_32.Position = [744 11 398 100];

            % Create CreateMakefileButton
            app.CreateMakefileButton = uibutton(app.Panel_32, 'push');
            app.CreateMakefileButton.ButtonPushedFcn = createCallbackFcn(app,
@CreateMakefileButtonPushed, true);
            app.CreateMakefileButton.VerticalAlignment = 'top';
            app.CreateMakefileButton.BackgroundColor = [0.9216 0.9216 0.9216];
            app.CreateMakefileButton.FontWeight = 'bold';
            app.CreateMakefileButton.Enable = 'off';
            app.CreateMakefileButton.Tooltip = {''};
            app.CreateMakefileButton.Position = [277 40 106 22];
            app.CreateMakefileButton.Text = 'Create Makefile';

            % Create StartsimulationButton
            app.StartsimulationButton = uibutton(app.Panel_32, 'push');
            app.StartsimulationButton.ButtonPushedFcn = createCallbackFcn(app, @StartsimulationButtonPushed, true);
            app.StartsimulationButton.BackgroundColor = [0.9216 0.9216 0.9216];
            app.StartsimulationButton.FontWeight = 'bold';
            app.StartsimulationButton.Enable = 'off';
            app.StartsimulationButton.Tooltip = {'Compile the Makefile to create ''funwave.exe'' and start the tsunami simulation.'};
            app.StartsimulationButton.Position = [277 14 107 22];
            app.StartsimulationButton.Text = 'Start simulation';

            % Create Button_42
            app.Button_42 = uibutton(app.Panel_32, 'push');
            app.Button_42.ButtonPushedFcn = createCallbackFcn(app, @Button_42Pushed2, true);
            app.Button_42.VerticalAlignment = 'top';
            app.Button_42.FontSize = 10;
            app.Button_42.FontAngle = 'italic';
            app.Button_42.Enable = 'off';
            app.Button_42.Tooltip = {''};
            app.Button_42.Position = [250 41 19 18];
            app.Button_42.Text = '...';

            % Create MakefileEditFieldLabel
            app.MakefileEditFieldLabel = uilabel(app.Panel_32);
            app.MakefileEditFieldLabel.Enable = 'off';
            app.MakefileEditFieldLabel.Tooltip = {'Specify the folder directory containing all the .o, .F, and .f90 files'};
            app.MakefileEditFieldLabel.Position = [15 39 106 22];
            app.MakefileEditFieldLabel.Text = 'Source folder';

            % Create MakefileEditField
            app.MakefileEditField = uieditfield(app.Panel_32, 'text');
            app.MakefileEditField.FontSize = 10;
            app.MakefileEditField.FontColor = [0.502 0.502 0.502];
            app.MakefileEditField.Enable = 'off';
            app.MakefileEditField.Tooltip = {''};
            app.MakefileEditField.Placeholder = 'FUNWAVE-TVD-Version_3.6/src';
            app.MakefileEditField.Position = [94 42 154 16];

            % Create ParallelModeCheckBox
            app.ParallelModeCheckBox = uicheckbox(app.Panel_32);
            app.ParallelModeCheckBox.Enable = 'off';
            app.ParallelModeCheckBox.Tooltip = {'Use parallel mode'; ''; 'Unchecked box - serial mode'};
            app.ParallelModeCheckBox.Text = 'Parallel Mode';
            app.ParallelModeCheckBox.Position = [158 66 95 22];

            % Create IntelCompilerCheckBox
            app.IntelCompilerCheckBox = uicheckbox(app.Panel_32);
            app.IntelCompilerCheckBox.Enable = 'off';
            app.IntelCompilerCheckBox.Tooltip = {'Use intel compiler'};
            app.IntelCompilerCheckBox.Text = 'Intel Compiler';
            app.IntelCompilerCheckBox.Position = [271 66 97 22];

            % Create DoublePrecisionCheckBox
            app.DoublePrecisionCheckBox = uicheckbox(app.Panel_32);
            app.DoublePrecisionCheckBox.ValueChangedFcn = createCallbackFcn(app, @DoublePrecisionCheckBoxValueChanged, true);
            app.DoublePrecisionCheckBox.Enable = 'off';
            app.DoublePrecisionCheckBox.Tooltip = {'Use double precision'; ''; 'Unchecked box - single precision'};
            app.DoublePrecisionCheckBox.Text = 'Double Precision';
            app.DoublePrecisionCheckBox.Position = [26 66 113 22];

            % Create Mpif90EditFieldLabel
            app.Mpif90EditFieldLabel = uilabel(app.Panel_32);
            app.Mpif90EditFieldLabel.Enable = 'off';
            app.Mpif90EditFieldLabel.Tooltip = {'Specify the directory containing the MPI F90 Compiler.'; ''; 'Mac OS default:'; '/opt/homebrew/bin/'; ''; 'Windows OS:'; 'C:\Program Files (x86)\MPICH2\bin'};
            app.Mpif90EditFieldLabel.Position = [15 13 77 22];
            app.Mpif90EditFieldLabel.Text = 'Mpif90 folder';

            % Create Mpif90EditField
            app.Mpif90EditField = uieditfield(app.Panel_32, 'text');
            app.Mpif90EditField.ValueChangedFcn = createCallbackFcn(app, @Mpif90EditFieldValueChanged, true);
            app.Mpif90EditField.ValueChangingFcn = createCallbackFcn(app, @Mpif90EditFieldValueChanging, true);
            app.Mpif90EditField.FontSize = 11;
            app.Mpif90EditField.FontColor = [0.502 0.502 0.502];
            app.Mpif90EditField.Enable = 'off';
            app.Mpif90EditField.Tooltip = {''};
            app.Mpif90EditField.Placeholder = 'C:\Program Files (x86)\MPICH2\bin\';
            app.Mpif90EditField.Position = [94 16 154 16];

            % Create Button_41
            app.Button_41 = uibutton(app.Panel_32, 'push');
            app.Button_41.ButtonPushedFcn = createCallbackFcn(app, @Button_41Pushed, true);
            app.Button_41.VerticalAlignment = 'top';
            app.Button_41.FontAngle = 'italic';
            app.Button_41.Enable = 'off';
            app.Button_41.Tooltip = {''};
            app.Button_41.Position = [250 15 19 19];
            app.Button_41.Text = '...';

            % Create Friction________________________________Label_9
            app.Friction________________________________Label_9 = uilabel(app.UIFigure);
            app.Friction________________________________Label_9.BackgroundColor = [0.9412 0.9412 0.9412];
            app.Friction________________________________Label_9.FontWeight =
'bold';
            app.Friction________________________________Label_9.Tooltip = {'This section becomes enabled once Makefile-related sections on the left panel are completed.'};
            app.Friction________________________________Label_9.Position = [758 100 67 22];
            app.Friction________________________________Label_9.Text = ' Makefile  ';

            % 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_Input_exported

            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
```