Internship Report

On

Restructuring of the User Interface to be common for all modules

Submitted by

Ansari Mohammed Umair

Under the guidance of

Prof.Siddhartha Ghosh
Civil Engineering Department
IIT Bombay

and

Prof.Sunil Shetye
Senior Project Manager
FOSSEE

Under the Mentorship of

Deepthi Reddy
Project Research Associate

January 27, 2020

Acknowledgment

I would like to thank FOSSEE for providing me a platform to work on something I am very interested in. I am thankful to everyone who thought of having and involved in selection process based on screening tasks. I am grateful to be a part of team which promotes open source software.

I thank all the Osdag members, who are wonderful mentors and great team. I thank Deepthi Reddy (Project Research Associate), Sourabh Das (Project Research Associate), Danish Ansari (Project Research Assistant), Yash Lokhande (Project Research Assistant), Anand Swaroop (Project Research Associate), Darshan Viswakarma (Project Research Associate), Anjali Jatav (Project Research Assistant) and whole team, who made us feel welcome and planned all the tasks meticulously during this period.

I am grateful that I got a chance to work under Prof. Siddhartha Ghosh, who took time to mentor us and monitored individual contributions as well.

Contents

Chapter 1

Introduction

1.1 Osdag Internship

Osdag internship is provided under the FOSSEE project. FOSSEE project promotes the use of FOSS (Free/Libre and Open Source Software) tools to improve quality of education in our country. FOSSEE encourages the use of FOSS tools through various activities to ensure availability of competent free software equivalent to commercial (paid) softwares.
The FOSSEE project is a part of the National Mission on Education through Infrastructure and Communication Technology(ICT), Ministry of Human Resources and Development, Government of India.
Osdag is one such open source software which comes under the FOSSEE project. Osdag internship is provided through FOSSEE project. Any UG/PG/PhD holder can apply for this internship. And the selection will be based on a screening task.

1.2 What is Osdag?

Osdag is Free/Libre and Open Source Software being developed for design of steel structures. Its source code is written in Python, 3D CAD images are developed using PythonOCC. Github is used to ensure smooth workflow between different modules and team members. It is in a path where people from around the world would be able to contribute to its development. FOSSEE's "Share alike" policy would improve the standard of the software when the source code is further modified based on the industrial and educational needs across the country.

1.3 Who can use ?

Osdag is created both for educational purpose and industry professionals. As Osdag is currently funded by MHRD, Osdag team is developing software in such a way that it can be used by the students during their academics and to give them a better insight look in the subject.
Osdag can be used by anyone starting from novice to professionals. It's simple user interface makes it flexible and attractive than other software. Video tutorials are available to help get started. The video tutorials of Osdag can be accessed here.

Chapter 2

Coding in Python for restructuring of the UI

I have written python code for restructuring of the connection module by using PyQt5 and specific functions for different modules which was previously generated by Qt designer.

Figure 2.1: Osdag-mainpage

2.1 Code for Input Dock

A function input_values is used to give the list of elements of input dock for every module which is then implemented in the UI using Qt Widgets. Concerned code is attached vide Appendix -A.

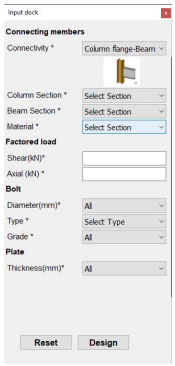

Figure 2.2: Input Dock

2.2 Code for Change in Input Dock based on Key-Connectivity

I have created a function to change the elements in input dock based on the values selected for key-connectivity. Concerned code is attached vide Appendix-B.

2.3 Code for Reset Button

I have created a function to reset the values of input dock on click of Reset Button. Concerned code is attached vide Appendix-C.

2.4 Code for Design Button

I have created a function to make a dictionary of the values of input dock entered by the user on click of Design Button. Concerned code is attached vide Appendix-D.

2.5 Code for Saving Design Inputs

I have created a function to save the inputs of a design in an osi file which can be used later as per requirement. Concerned code is attached vide Appendix-E.

2.6 Code to Loading Design Inputs

I have created a function to load the inputs from an osi file to the UI. Concerned code is attached vide Appendix-F.

2.7 Code for Design Preferences

I have created a function for design preferences for all connection modules which is used to edit the design preferences as per user requirements and new set of custom design preferences can be added to the database. Concerned code is attached vide Appendix-G.

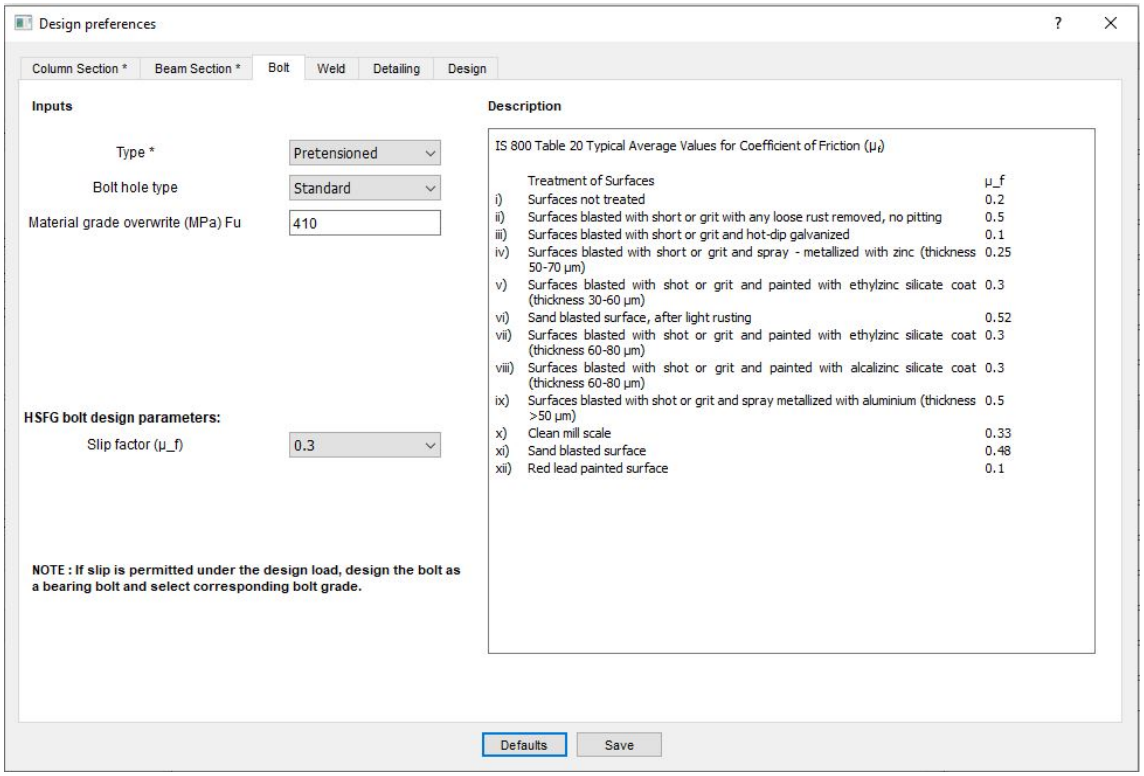

Figure 2.3: Design Preferences

2.8 Code for Output Dock

A function output_values is used to give the list of elements of output dock for every module which is then implemented in the UI using Qt Widgets. Concerned code is attached vide Appendix-H.

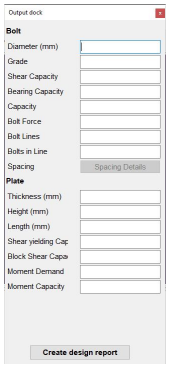

Figure 2.4: Output Dock

2.9 Code for Reloading values from database to input Dock

After successful addition of new custom values into the database by user, the values are reloaded into the UI from database so that users can directly access their custom input values for design. Concerned code is attached vide Appendix-I.

2.10 Code for Dialog Box in Output Dock

On click of a button in output dock, a dialog box is shown which contains certain set of results about the design. Concerned code is attached vide Appendix-J.

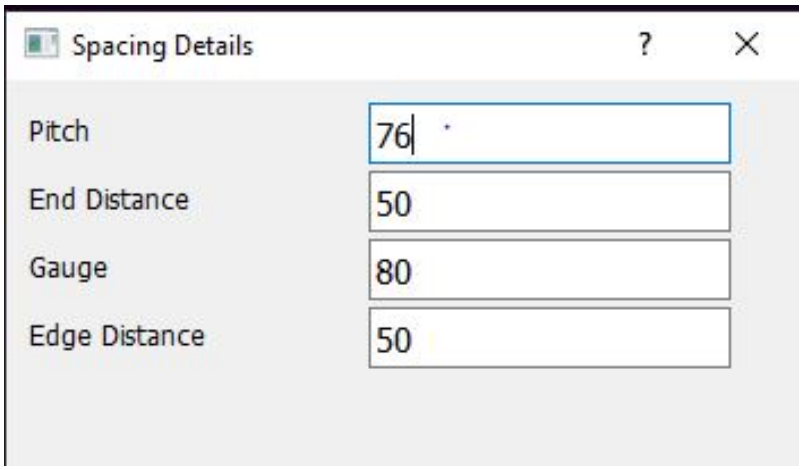

Figure 2.5: Dialog box in Output dock

Appendices

Appendix A

Code for Input Dock

```
512         option_list = main.input_values(self)
513         _translate = QtCore.QCoreApplication.translate
514 
515         i = 0
516         for option in option_list:
517             lable = option[1]
518             type = option[2]
519             if type not in [TYPE_TITLE, TYPE_IMAGE, TYPE_MODULE]:
520                 l = QtWidgets.QLabel(self.dockWidgetContents)
521                 l.setGeometry(QtCore.QRect(6, 10 + i, 120, 25))
522                 font = QtGui.QFont()
523                 font.setPointSize(11)
524                 font.setBold(False)
525                 font.setWeight(50)
526                 l.setFont(font)
527                 l.setObjectName(option[0] + "_label")
528                 l.setText(_translate("MainWindow", "<html><head/><body><p>" +
                    ↪  lable + "</p></body></html>"))
529 
530             if type == TYPE_COMBOBOX or type == TYPE_COMBOBOX_CUSTOMIZED:
531                 combo = QtWidgets.QComboBox(self.dockWidgetContents)
532                 combo.setGeometry(QtCore.QRect(150, 10 + i, 160, 27))
533                 font = QtGui.QFont()
534                 font.setPointSize(11)
535                 font.setBold(False)
536                 font.setWeight(50)
537                 combo.setFont(font)
538                 combo.view().setVerticalScrollBarPolicy(Qt.ScrollBarAsNeeded)
539                 combo.setStyleSheet("QComboBox { combobox-popup: 0; }")
540                 combo.setMaxVisibleItems(5)
541                 combo.setObjectName(option[0])
542                 for item in option[4]:
543                     combo.addItem(item)
544 
545             if type == TYPE_TEXTBOX:
546                 r = QtWidgets.QLineEdit(self.dockWidgetContents)
547                 r.setGeometry(QtCore.QRect(150, 10 + i, 160, 27))
548                 font = QtGui.QFont()
549                 font.setPointSize(11)
550                 font.setBold(False)
```

```
                font.setWeight(50)
                r.setFont(font)
                r.setObjectName(option[0])

            if type == TYPE_MODULE:
                _translate = QtCore.QCoreApplication.translate
                MainWindow.setWindowTitle(_translate("MainWindow", option[1]))
                i = i - 30
                module = lable

            if type == TYPE_IMAGE:
                im = QtWidgets.QLabel(self.dockWidgetContents)
                im.setGeometry(QtCore.QRect(190, 10 + i, 70, 57))
                im.setObjectName(option[0])
                im.setScaledContents(True)
                pixmap = QPixmap("./ResourceFiles/images/fin_cf_bw.png")
                im.setPixmap(pixmap)
                i = i + 30

            if option[0] in [KEY_AXIAL, KEY_SHEAR]:
                key = self.dockWidgetContents.findChild(QtWidgets.QWidget,
                ↪  option[0])
                onlyInt = QIntValidator()
                key.setValidator(onlyInt)

            if type == TYPE_TITLE:
                q = QtWidgets.QLabel(self.dockWidgetContents)
                q.setGeometry(QtCore.QRect(3, 10 + i, 201, 25))
                font = QtGui.QFont()
                q.setFont(font)
                q.setObjectName("_title")
                q.setText(_translate("MainWindow",
                                     "<html><head/><body><p><span style=\"
                                     ↪  font-weight:600;\">" + lable +
                                     ↪  "</span></p></body></html>"))
            i = i + 30
```

```

    def input_values(self, existingvalues={}):

        '''
        Fuction to return a list of tuples to be displayed as the UI.(Input
↪   Dock)
        '''

        # @author: Amir, Umair

        options_list = []

        if KEY_CONN in existingvalues:
            existingvalue_key_conn = existingvalues[KEY_CONN]
        else:
            existingvalue_key_conn = ''

        if KEY_SUPTNGSEC in existingvalues:
```

```
            existingvalue_key_suptngsec = existingvalues[KEY_SUPTNGSEC]
        else:
            existingvalue_key_suptngsec = ''

        if KEY_SUPTDSEC in existingvalues:
            existingvalue_key_suptdsec = existingvalues[KEY_SUPTDSEC]
        else:
            existingvalue_key_suptdsec = ''

        if KEY_MATERIAL in existingvalues:
            existingvalue_key_mtrl = existingvalues[KEY_MATERIAL]
        else:
            existingvalue_key_mtrl = ''

        if KEY_SHEAR in existingvalues:
            existingvalue_key_versh = existingvalues[KEY_SHEAR]
        else:
            existingvalue_key_versh = ''

        if KEY_AXIAL in existingvalues:
            existingvalue_key_axial = existingvalues[KEY_AXIAL]
        else:
            existingvalue_key_axial = ''

        if KEY_D in existingvalues:
            existingvalue_key_d = existingvalues[KEY_D]
        else:
            existingvalue_key_d = ''

        if KEY_TYP in existingvalues:
            existingvalue_key_typ = existingvalues[KEY_TYP]
        else:
            existingvalue_key_typ = ''

        if KEY_GRD in existingvalues:
            existingvalue_key_grd = existingvalues[KEY_GRD]
        else:
            existingvalue_key_grd = ''

        if KEY_PLATETHK in existingvalues:
            existingvalue_key_platethk = existingvalues[KEY_PLATETHK]
        else:
            existingvalue_key_platethk = ''


        t16 = (KEY_MODULE, KEY_DISP_FINPLATE, TYPE_MODULE, None, None)
        options_list.append(t16)

        t1 = (None, DISP_TITLE_CM, TYPE_TITLE, None, None)
        options_list.append(t1)

        t2 = (KEY_CONN, KEY_DISP_CONN, TYPE_COMBOBOX, existingvalue_key_conn,
        ↪  VALUES_CONN)
        options_list.append(t2)

        t15 = (KEY_IMAGE, None, TYPE_IMAGE, None, None)
```

```
157         options_list.append(t15)
158
159         t3 = (KEY_SUPTNGSEC, KEY_DISP_COLSEC, TYPE_COMBOBOX,
            ↪  existingvalue_key_suptngsec, connectdb("Columns"))
160         options_list.append(t3)
161
162         t4 = (KEY_SUPTDSEC, KEY_DISP_BEAMSEC, TYPE_COMBOBOX,
            ↪  existingvalue_key_suptdsec, connectdb("Beams"))
163         options_list.append(t4)
164
165         t5 = (KEY_MATERIAL, KEY_DISP_MATERIAL, TYPE_COMBOBOX,
            ↪  existingvalue_key_mtrl, VALUES_MATERIAL)
166         options_list.append(t5)
167
168         t6 = (None, DISP_TITLE_FSL, TYPE_TITLE, None, None)
169         options_list.append(t6)
170
171         t7 = (KEY_SHEAR, KEY_DISP_SHEAR, TYPE_TEXTBOX, existingvalue_key_versh,
            ↪  None)
172         options_list.append(t7)
173
174         t8 = (KEY_AXIAL, KEY_DISP_AXIAL, TYPE_TEXTBOX, existingvalue_key_axial,
            ↪  None)
175         options_list.append(t8)
176
177         t9 = (None, DISP_TITLE_BOLT, TYPE_TITLE, None, None)
178         options_list.append(t9)
179
180         t10 = (KEY_D, KEY_DISP_D, TYPE_COMBOBOX_CUSTOMIZED,
            ↪  existingvalue_key_d, VALUES_D)
181         options_list.append(t10)
182
183         t11 = (KEY_TYP, KEY_DISP_TYP, TYPE_COMBOBOX, existingvalue_key_typ,
            ↪  VALUES_TYP)
184         options_list.append(t11)
185
186         t12 = (KEY_GRD, KEY_DISP_GRD, TYPE_COMBOBOX_CUSTOMIZED,
            ↪  existingvalue_key_grd, VALUES_GRD)
187         options_list.append(t12)
188
189         t13 = (None, DISP_TITLE_PLATE, TYPE_TITLE, None, None)
190         options_list.append(t13)
191
192         t14 = (KEY_PLATETHK, KEY_DISP_PLATETHK, TYPE_COMBOBOX_CUSTOMIZED,
            ↪  existingvalue_key_platethk, VALUES_PLATETHK)
193         options_list.append(t14)
194
195         return options_list
196
```

Appendix B

Code for Change on Key-Connectivity

```
645         updated_list = main.input_value_changed(main)
646         if updated_list is None:
647             pass
648         else:
649             for t in updated_list:
650                 key_changed =
                    ↪  self.dockWidgetContents.findChild(QtWidgets.QWidget, t[0])
651                 key_changed.currentIndexChanged.connect(lambda:
                    ↪  change(key_changed, updated_list))
652
653         def change(k1, new):
654
655             """
656             @author: Umair
657             """
658
659             for tup in new:
660                 (object_name, k2_key, typ, f) = tup
661                 if object_name != k1.objectName():
662                     continue
663                 if typ == TYPE_LABEL:
664                     k2_key = k2_key + "_label"
665                 k2 = self.dockWidgetContents.findChild(QtWidgets.QWidget,
                    ↪  k2_key)
666                 val = f(k1.currentText())
667                 k2.clear()
668                 if typ == TYPE_COMBOBOX:
669                     for values in val:
670                         k2.addItem(values)
671                         k2.setCurrentIndex(0)
672                     if k2_key in [KEY_SUPTNGSEC, KEY_SUPTDSEC, KEY_SECSIZE]:
673                         red_list_set = set(red_list_function())
674                         current_list_set = set(val)
675                         current_red_list =
                            ↪  list(current_list_set.intersection(red_list_set))
676                         for value in current_red_list:
677                             indx = val.index(str(value))
```

```
678                             k2.setItemData(indx, QBrush(QColor("red")),
                                ↪  Qt.TextColorRole)
679                 elif typ == TYPE_LABEL:
680                     k2.setText(val)
681                 elif typ == TYPE_IMAGE:
682                     pixmap1 = QPixmap(val)
683                     k2.setPixmap(pixmap1)
684                 else:
685                     pass
686
```

Appendix C

Code for Reset Button

```
1187     def reset_fn(self, op_list, out_list, new_list, data):
1188 
1189         # For input dock
1190 
1191         for op in op_list:
1192             widget = self.dockWidgetContents.findChild(QtWidgets.QWidget,
                 ↪  op[0])
1193             if op[2] == TYPE_COMBOBOX or op[2] == TYPE_COMBOBOX_CUSTOMIZED:
1194                 widget.setCurrentIndex(0)
1195             elif op[2] == TYPE_TEXTBOX:
1196                 widget.setText('')
1197             else:
1198                 pass
1199 
1200         # For list in Customized combobox
1201 
1202         for custom_combo in new_list:
1203             data[custom_combo[0] + "_customized"] = custom_combo[1]()
1204 
1205         # For output dock
1206 
1207         for out in out_list:
1208             widget = self.dockWidgetContents_out.findChild(QtWidgets.QWidget,
                 ↪  out[0])
1209             if out[2] == TYPE_TEXTBOX:
1210                 widget.setText('')
1211             else:
1212                 pass
1213 
```

Appendix D

Code for Design Button

```
1219     def design_fn(self, op_list, data_list):
1220         design_dictionary = {}
1221         for op in op_list:
1222             widget = self.dockWidgetContents.findChild(QtWidgets.QWidget,
                 ↪  op[0])
1223             if op[2] == TYPE_COMBOBOX:
1224                 des_val = widget.currentText()
1225                 d1 = {op[0]: des_val}
1226             elif op[2] == TYPE_MODULE:
1227                 des_val = op[1]
1228                 d1 = {op[0]: des_val}
1229             elif op[2] == TYPE_COMBOBOX_CUSTOMIZED:
1230                 des_val = data_list[op[0]+"_customized"]
1231                 d1 = {op[0]: des_val}
1232             elif op[2] == TYPE_TEXTBOX:
1233                 des_val = widget.text()
1234                 d1 = {op[0]: des_val}
1235             else:
1236                 d1 = {}
1237             design_dictionary.update(d1)
1238         design_dictionary.update(self.designPrefDialog.save_designPref_para())
1239         self.design_inputs = design_dictionary
1240
```

Appendix E

Code for Saving Design Inputs

```
1253     '''
1254     @author: Umair
1255     '''
1256     def saveDesign_inputs(self):
1257         fileName, _ = QFileDialog.getSaveFileName(self,
1258                                                   "Save Design", os.path.join('
                                                      ↪  ', "untitled.osi"),
1259                                                   "Input Files(*.osi)")
1260         if not fileName:
1261             return
1262         try:
1263             with open(fileName, 'w') as input_file:
1264                 yaml.dump(self.design_inputs, input_file)
1265         except Exception as e:
1266             QMessageBox.warning(self, "Application",
1267                                 "Cannot write file %s:\n%s" % (fileName,
                                    ↪  str(e)))
1268             return
1269
```

Appendix F

Code for Loading Design Inputs

```
1274
1275        def loadDesign_inputs(self, op_list, data, new):
1276            fileName, _ = QFileDialog.getOpenFileName(self, "Open Design",
                ↪  os.path.join(str(' '), ''), "InputFiles(*.osi)")
1277            if not fileName:
1278                return
1279            try:
1280                in_file = str(fileName)
1281                with open(in_file, 'r') as fileObject:
1282                    uiObj = yaml.load(fileObject)
1283                self.setDictToUserInputs(uiObj, op_list, data, new)
1284
1285            except IOError:
1286                QMessageBox.information(self, "Unable to open file",
1287                                        "There was an error opening \"%s\"" %
                                            ↪  fileName)
1288                return
1289
1290    # Function for loading inputs from a file to Ui
1291        '''
1292        @author: Umair
1293        '''
1294
1295        def setDictToUserInputs(self, uiObj, op_list, data, new):
1296            for op in op_list:
1297                key_str = op[0]
1298                key = self.dockWidgetContents.findChild(QtWidgets.QWidget, key_str)
1299                if op[2] == TYPE_COMBOBOX:
1300                    index = key.findText(uiObj[key_str],
                        ↪  QtCore.Qt.MatchFixedString)
1301                    if index >= 0:
1302                        key.setCurrentIndex(index)
1303                elif op[2] == TYPE_TEXTBOX:
1304                    key.setText(uiObj[key_str])
1305                elif op[2] == TYPE_COMBOBOX_CUSTOMIZED:
1306                    for n in new:
1307                        if n[0] == key_str:
1308                            if uiObj[key_str] != n[1]():
1309                                data[key_str + "_customized"] = uiObj[key_str]
1310                                key.setCurrentIndex(1)
```

Appendix G

Code for Design Preferences

```
1742
1743        def combined_design_prefer(self, module):
1744            key_1 = self.dockWidgetContents.findChild(QtWidgets.QWidget, KEY_CONN)
1745            key_2 = self.dockWidgetContents.findChild(QtWidgets.QWidget,
                ↪  KEY_SUPTNGSEC)
1746            key_3 = self.dockWidgetContents.findChild(QtWidgets.QWidget,
                ↪  KEY_SUPTDSEC)
1747            key_4 = self.dockWidgetContents.findChild(QtWidgets.QWidget,
                ↪  KEY_MATERIAL)
1748            key_5 = self.dockWidgetContents.findChild(QtWidgets.QWidget,
                ↪  KEY_SECSIZE)
1749
1750            table_1 = "Columns"
1751            table_2 = "Beams"
1752            if module == KEY_DISP_BEAMCOVERPLATE:
1753                t = table_2
1754            elif module == KEY_DISP_COLUMNCOVERPLATE:
1755                t = table_1
1756            material_grade = key_4.currentText()
1757            if module in [KEY_DISP_BEAMCOVERPLATE, KEY_DISP_COLUMNCOVERPLATE]:
1758                designation_col = key_5.currentText()
1759                self.designPrefDialog.ui.tabWidget.removeTab(
1760                    self.designPrefDialog.ui.tabWidget.indexOf(
1761                        self.designPrefDialog.ui.tab_Beam))
1762                self.designPrefDialog.ui.tabWidget.setTabText(
1763                    self.designPrefDialog.ui.tabWidget.indexOf(
1764                    self.designPrefDialog.ui.tab_Column), KEY_DISP_SECSIZE)
1765                if key_5.currentIndex() != 0:
1766                    self.designPrefDialog.column_preferences(designation_col, t,
                        ↪  material_grade)
1767            else:
1768                conn = key_1.currentText()
1769                designation_col = key_2.currentText()
1770                designation_bm = key_3.currentText()
1771                if conn in VALUES_CONN_1:
1772                    self.designPrefDialog.ui.tabWidget.setTabText(
1773                        self.designPrefDialog.ui.tabWidget.indexOf(
1774                        self.designPrefDialog.ui.tab_Column), KEY_DISP_COLSEC)
1775                    self.designPrefDialog.ui.tabWidget.setTabText(
1776                        self.designPrefDialog.ui.tabWidget.indexOf(
```

```
1777                     self.designPrefDialog.ui.tab_Beam), KEY_DISP_BEAMSEC)
1778                 self.designPrefDialog.column_preferences(designation_col,
                     ↪  table_1, material_grade)
1779                 self.designPrefDialog.beam_preferences(designation_bm,
                     ↪  material_grade)
1780             elif conn in VALUES_CONN_2:
1781                 self.designPrefDialog.ui.tabWidget.setTabText(
1782                     self.designPrefDialog.ui.tabWidget.indexOf(
1783                     self.designPrefDialog.ui.tab_Column), KEY_DISP_PRIBM)
1784                 self.designPrefDialog.ui.tabWidget.setTabText(
1785                     self.designPrefDialog.ui.tabWidget.indexOf(
1786                     self.designPrefDialog.ui.tab_Beam), KEY_DISP_SECBM)
1787                 self.designPrefDialog.column_preferences(designation_col,
                     ↪  table_2, material_grade)
1788                 self.designPrefDialog.beam_preferences(designation_bm,
                     ↪  material_grade)
1789
```

```
723         pushButton_Clear_Column.clicked.connect(lambda:
            ↪  self.clear_tab("Column"))
724         pushButton_Clear_Beam.clicked.connect(lambda: self.clear_tab("Beam"))
725
726         pushButton_Add_Column.clicked.connect(self.add_tab_column)
727         pushButton_Add_Beam.clicked.connect(self.add_tab_beam)
728
729     def clear_tab(self, tab_name):
730         '''
731         @author: Umair
732         '''
733         if tab_name == "Column":
734             tab = self.tab_Column
735         elif tab_name == "Beam":
736             tab = self.tab_Beam
737         for c in tab.children():
738             if isinstance(c, QtWidgets.QComboBox):
739                 c.setCurrentIndex(0)
740             elif isinstance(c, QtWidgets.QLineEdit):
741                 c.clear()
742
743     def add_tab_column(self):
744         '''
745         @author: Umair
746         '''
747         name = self.tabWidget.tabText(self.tabWidget.indexOf(self.tab_Column))
748         if name == KEY_DISP_COLSEC:
749             table = "Columns"
750         elif name in [KEY_DISP_PRIBM, KEY_DISP_SECSIZE]:
751             table = "Beams"
752         else:
753             pass
754
755         for ch in self.tab_Column.children():
756             if isinstance(ch, QtWidgets.QLineEdit) and ch.text() == "":
757                 QMessageBox.information(QMessageBox(), 'Warning', 'Please Fill
                    ↪  all missing parameters!')
```

```
758                 add_col = self.tab_Column.findChild(QtWidgets.QWidget,
                    ↪  'pushButton_Add_Column')
759                 add_col.setDisabled(True)
760                 break
761             elif isinstance(ch, QtWidgets.QLineEdit) and ch.text() != "":
762                 if ch.objectName() == KEY_SUPTNGSEC_DESIGNATION:
763                     Designation_c = ch.text()
764                 elif ch.objectName() == KEY_SUPTNGSEC_SOURCE:
765                     Source_c = ch.text()
766                 elif ch.objectName() == KEY_SUPTNGSEC_DEPTH:
767                     D_c = float(ch.text())
768                 elif ch.objectName() == KEY_SUPTNGSEC_FLANGE_W:
769                     B_c = float(ch.text())
770                 elif ch.objectName() == KEY_SUPTNGSEC_FLANGE_T:
771                     T_c = float(ch.text())
772                 elif ch.objectName() == KEY_SUPTNGSEC_WEB_T:
773                     tw_c = float(ch.text())
774                 elif ch.objectName() == KEY_SUPTNGSEC_FLANGE_S:
775                     FlangeSlope_c = float(ch.text())
776                 elif ch.objectName() == KEY_SUPTNGSEC_ROOT_R:
777                     R1_c = float(ch.text())
778                 elif ch.objectName() == KEY_SUPTNGSEC_TOE_R:
779                     R2_c = float(ch.text())
780                 elif ch.objectName() == KEY_SUPTNGSEC_MASS:
781                     Mass_c = float(ch.text())
782                 elif ch.objectName() == KEY_SUPTNGSEC_SEC_AREA:
783                     Area_c = float(ch.text())
784                 elif ch.objectName() == KEY_SUPTNGSEC_MOA_LZ:
785                     Iz_c = float(ch.text())
786                 elif ch.objectName() == KEY_SUPTNGSEC_MOA_LY:
787                     Iy_c = float(ch.text())
788                 elif ch.objectName() == KEY_SUPTNGSEC_ROG_RZ:
789                     rz_c = float(ch.text())
790                 elif ch.objectName() == KEY_SUPTNGSEC_ROG_RY:
791                     ry_c = float(ch.text())
792                 elif ch.objectName() == KEY_SUPTNGSEC_EM_ZZ:
793                     Zz_c = float(ch.text())
794                 elif ch.objectName() == KEY_SUPTNGSEC_EM_ZY:
795                     Zy_c = float(ch.text())
796                 elif ch.objectName() == KEY_SUPTNGSEC_PM_ZPZ:
797                     if ch.text() == "":
798                         ch.setText("0")
799                     Zpz_c = ch.text()
800                 elif ch.objectName() == KEY_SUPTNGSEC_PM_ZPY:
801                     if ch.text() == "":
802                         ch.setText("0")
803                     Zpy_c = ch.text()
804                 else:
805                     pass
806             elif isinstance(ch, QtWidgets.QComboBox):
807                 if ch.objectName() == KEY_SUPTNGSEC_TYPE:
808                     Type = ch.currentText()
809 
810         if ch == self.tab_Column.children()[len(self.tab_Column.children())-1]:
811             conn = sqlite3.connect(PATH_TO_DATABASE)
812             c = conn.cursor()
```

```
813             if table == "Beams":
814                 c.execute("SELECT count(*) FROM Beams WHERE Designation = ?",
                    ↪  (Designation_c,))
815                 data = c.fetchone()[0]
816             else:
817                 c.execute("SELECT count(*) FROM Columns WHERE Designation = ?",
                    ↪  (Designation_c,))
818                 data = c.fetchone()[0]
819             if data == 0:
820                 if table == "Beams":
821                     c.execute('''INSERT INTO Beams
                        ↪  (Designation,Mass,Area,D,B,tw,T,R1,R2,Iz,Iy,rz,ry,
822                         Zz,zy,Zpz,Zpy,FlangeSlope,Source,Type) VALUES
    ↪  (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)''',
823                                 (Designation_c, Mass_c, Area_c,
824                                  D_c, B_c, tw_c, T_c,
825                                  R1_c, R2_c, Iz_c, Iy_c, rz_c,
826                                  ry_c, Zz_c, Zy_c,
827                                  Zpz_c, Zpy_c, FlangeSlope_c, Source_c, Type))
828                     conn.commit()
829                 else:
830                     c.execute('''INSERT INTO Columns
                        ↪  (Designation,Mass,Area,D,B,tw,T,R1,R2,Iz,Iy,rz,ry,
831                         Zz,zy,Zpz,Zpy,FlangeSlope,Source,Type) VALUES
    ↪  (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)''',
832                                 (Designation_c, Mass_c, Area_c,
833                                  D_c, B_c, tw_c, T_c,
834                                  R1_c, R2_c, Iz_c, Iy_c, rz_c,
835                                  ry_c, Zz_c, Zy_c,
836                                  Zpz_c, Zpy_c, FlangeSlope_c, Source_c, Type))
837                     conn.commit()
838                 c.close()
839                 conn.close()
840                 QMessageBox.information(QMessageBox(), 'Information', 'Data is
                    ↪  added successfully to the database!')
841 
842             else:
843                 QMessageBox.information(QMessageBox(), 'Warning', 'Designation
                    ↪  is already exist in Database!')
844 
845     def add_tab_beam(self):
846         '''
847         @author: Umair
848         '''
849 
850         for ch in self.tab_Beam.children():
851 
852             if isinstance(ch, QtWidgets.QLineEdit) and ch.text() == "":
853                 QMessageBox.information(QMessageBox(), 'Warning', 'Please Fill
                    ↪  all missing parameters!')
854                 add_bm = self.tab_Beam.findChild(QtWidgets.QWidget,
                    ↪  'pushButton_Add_Beam')
855                 add_bm.setDisabled(True)
856                 break
857 
858             elif isinstance(ch, QtWidgets.QLineEdit) and ch.text() != "":
```

```
859
860                 if ch.objectName() == KEY_SUPTDSEC_DESIGNATION:
861                     Designation_b = ch.text()
862                 elif ch.objectName() == KEY_SUPTDSEC_SOURCE:
863                     Source_b = ch.text()
864                 elif ch.objectName() == KEY_SUPTDSEC_DEPTH:
865                     D_b = float(ch.text())
866                 elif ch.objectName() == KEY_SUPTDSEC_FLANGE_W:
867                     B_b = float(ch.text())
868                 elif ch.objectName() == KEY_SUPTDSEC_FLANGE_T:
869                     T_b = float(ch.text())
870                 elif ch.objectName() == KEY_SUPTDSEC_WEB_T:
871                     tw_b = float(ch.text())
872                 elif ch.objectName() == KEY_SUPTDSEC_FLANGE_S:
873                     FlangeSlope_b = float(ch.text())
874                 elif ch.objectName() == KEY_SUPTDSEC_ROOT_R:
875                     R1_b = float(ch.text())
876                 elif ch.objectName() == KEY_SUPTDSEC_TOE_R:
877                     R2_b = float(ch.text())
878                 elif ch.objectName() == KEY_SUPTDSEC_MASS:
879                     Mass_b = float(ch.text())
880                 elif ch.objectName() == KEY_SUPTDSEC_SEC_AREA:
881                     Area_b = float(ch.text())
882                 elif ch.objectName() == KEY_SUPTDSEC_MOA_LZ:
883                     Iz_b = float(ch.text())
884                 elif ch.objectName() == KEY_SUPTDSEC_MOA_LY:
885                     Iy_b = float(ch.text())
886                 elif ch.objectName() == KEY_SUPTDSEC_ROG_RZ:
887                     rz_b = float(ch.text())
888                 elif ch.objectName() == KEY_SUPTDSEC_ROG_RY:
889                     ry_b = float(ch.text())
890                 elif ch.objectName() == KEY_SUPTDSEC_EM_ZZ:
891                     Zz_b = float(ch.text())
892                 elif ch.objectName() == KEY_SUPTDSEC_EM_ZY:
893                     Zy_b = float(ch.text())
894                 elif ch.objectName() == KEY_SUPTDSEC_PM_ZPZ:
895                     if ch.text() == "":
896                         ch.setText("0")
897                     Zpz_b = ch.text()
898                 elif ch.objectName() == KEY_SUPTDSEC_PM_ZPY:
899                     if ch.text() == "":
900                         ch.setText("0")
901                     Zpy_b = ch.text()
902                 else:
903                     pass
904             elif isinstance(ch, QtWidgets.QComboBox):
905                 if ch.objectName() == KEY_SUPTDSEC_TYPE:
906                     Type = ch.currentText()
907
908         if ch == self.tab_Beam.children()[len(self.tab_Beam.children())-1]:
909
910             conn = sqlite3.connect(PATH_TO_DATABASE)
911
912             c = conn.cursor()
913             c.execute("SELECT count(*) FROM Beams WHERE Designation = ?",
                ↪  (Designation_b,))
```

```
914             data = c.fetchone()[0]
915             if data == 0:
916                 c.execute('''INSERT INTO Beams
                    ↪  (Designation,Mass,Area,D,B,tw,T,R1,R2,Iz,Iy,rz,ry,
917                     Zz,zy,Zpz,Zpy,FlangeSlope,Source,Type) VALUES
918                     (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)''',
919                           (Designation_b, Mass_b, Area_b,
920                            D_b, B_b, tw_b, T_b, FlangeSlope_b,
921                            R1_b, R2_b, Iz_b, Iy_b, rz_b,
922                            ry_b, Zz_b, Zy_b,
923                            Zpz_b, Zpy_b, Source_b, Type))
924                 conn.commit()
925                 c.close()
926                 conn.close()
927                 QMessageBox.information(QMessageBox(), 'Information', 'Data is
                    ↪  added successfully to the database.')
928             else:
929                 QMessageBox.information(QMessageBox(), 'Warning', 'Designation
                    ↪  is already exist in Database!')
930
```

```
959
960     def default_fn(self):
961         '''
962         @author: Umair
963         '''
964         for children in self.ui.tab_Bolt.children():
965             if children.objectName() == KEY_DP_BOLT_TYPE:
966                 children.setCurrentIndex(0)
967             elif children.objectName() == KEY_DP_BOLT_HOLE_TYPE:
968                 children.setCurrentIndex(0)
969             elif children.objectName() == KEY_DP_BOLT_MATERIAL_G_O:
970                 children.setText('410')
971             elif children.objectName() == KEY_DP_BOLT_SLIP_FACTOR:
972                 children.setCurrentIndex(4)
973             else:
974                 pass
975         for children in self.ui.tab_Weld.children():
976             if children.objectName() == KEY_DP_WELD_TYPE:
977                 children.setCurrentIndex(0)
978             elif children.objectName() == KEY_DP_WELD_MATERIAL_G_O:
979                 children.setText('410')
980             else:
981                 pass
982         for children in self.ui.tab_Detailing.children():
983             if children.objectName() == KEY_DP_DETAILING_EDGE_TYPE:
984                 children.setCurrentIndex(0)
985             elif children.objectName() == KEY_DP_DETAILING_GAP:
986                 children.setText('10')
987             elif children.objectName() ==
                ↪  KEY_DP_DETAILING_CORROSIVE_INFLUENCES:
988                 children.setCurrentIndex(0)
989             else:
990                 pass
991         for children in self.ui.tab_Design.children():
```

```
            if children.objectName() == KEY_DP_DESIGN_METHOD:
                children.setCurrentIndex(0)
            else:
                pass


    def save_designPref_para(self):
        """This routine is responsible for saving all design preferences
        ↪  selected by the user
        """
        '''
        @author: Umair
        '''
        key_boltHoleType = self.ui.tab_Bolt.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_BOLT_HOLE_TYPE)
        combo_boltHoleType = key_boltHoleType.currentText()
        key_boltFu = self.ui.tab_Bolt.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_BOLT_MATERIAL_G_O)
        line_boltFu = key_boltFu.text()
        key_slipfactor = self.ui.tab_Bolt.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_BOLT_SLIP_FACTOR)
        combo_slipfactor = key_slipfactor.currentText()
        key_weldType = self.ui.tab_Weld.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_WELD_TYPE)
        combo_weldType = key_weldType.currentText()
        key_weldFu = self.ui.tab_Weld.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_WELD_MATERIAL_G_O)
        line_weldFu = key_weldFu.text()
        key_detailingEdgeType =
        ↪  self.ui.tab_Detailing.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_DETAILING_EDGE_TYPE)
        combo_detailingEdgeType = key_detailingEdgeType.currentText()
        key_detailingGap = self.ui.tab_Detailing.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_DETAILING_GAP)
        line_detailingGap = key_detailingGap.text()
        key_detailing_memebers =
        ↪  self.ui.tab_Detailing.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_DETAILING_CORROSIVE_INFLUENCES)
        combo_detailing_memebers = key_detailing_memebers.currentText()
        key_design_method = self.ui.tab_Design.findChild(QtWidgets.QWidget,
        ↪  KEY_DP_DESIGN_METHOD)
        combo_design_method = key_design_method.currentText()
        d1 = {KEY_DP_BOLT_HOLE_TYPE: combo_boltHoleType,
              KEY_DP_BOLT_MATERIAL_G_O: line_boltFu,
              KEY_DP_BOLT_SLIP_FACTOR: combo_slipfactor,
              KEY_DP_WELD_TYPE: combo_weldType,
              KEY_DP_WELD_MATERIAL_G_O: line_weldFu,
              KEY_DP_DETAILING_EDGE_TYPE: combo_detailingEdgeType,
              KEY_DP_DETAILING_GAP: line_detailingGap,
              KEY_DP_DETAILING_CORROSIVE_INFLUENCES: combo_detailing_memebers,
              ↪  KEY_DP_DESIGN_METHOD: combo_design_method}

        return d1
```

```
1087    def column_preferences(self, designation, table, material_grade):
1088        '''
1089        @author: Umair
1090        '''
1091
1092        if designation == 'Select Section':
1093            self.ui.clear_tab("Column")
1094            return
1095
1096        col_list = []
1097        col_attributes = Section(designation, material_grade)
1098        Section.connect_to_database_update_other_attributes(col_attributes,
          ↪  table, designation)
1099
1100        for ch in self.ui.tab_Column.children():
1101            if ch.objectName() == KEY_SUPTNGSEC_DESIGNATION:
1102                ch.setText(designation)
1103            elif ch.objectName() == KEY_SUPTNGSEC_SOURCE:
1104                ch.setText(col_attributes.source)
1105            elif ch.objectName() == KEY_SUPTNGSEC_FU:
1106                ch.setText(str(col_attributes.fu))
1107            elif ch.objectName() == KEY_SUPTNGSEC_FY:
1108                ch.setText(str(col_attributes.fy))
1109            elif ch.objectName() == KEY_SUPTNGSEC_DEPTH:
1110                ch.setText(str(col_attributes.depth))
1111                col_list.append(ch)
1112            elif ch.objectName() == KEY_SUPTNGSEC_FLANGE_W:
1113                ch.setText(str(col_attributes.flange_width))
1114                col_list.append(ch)
1115            elif ch.objectName() == KEY_SUPTNGSEC_FLANGE_T:
1116                ch.setText(str(col_attributes.flange_thickness))
1117                col_list.append(ch)
1118            elif ch.objectName() == KEY_SUPTNGSEC_WEB_T:
1119                ch.setText(str(col_attributes.web_thickness))
1120                col_list.append(ch)
1121            elif ch.objectName() == KEY_SUPTNGSEC_FLANGE_S:
1122                ch.setText(str(col_attributes.flange_slope))
1123            elif ch.objectName() == KEY_SUPTNGSEC_ROOT_R:
1124                ch.setText(str(col_attributes.root_radius))
1125            elif ch.objectName() == KEY_SUPTNGSEC_TOE_R:
1126                ch.setText(str(col_attributes.toe_radius))
1127            elif ch.objectName() == KEY_SUPTNGSEC_MOD_OF_ELAST:
1128                ch.setText("200")
1129                ch.setDisabled(True)
1130            elif ch.objectName() == KEY_SUPTNGSEC_MOD_OF_RIGID:
1131                ch.setText("76.9")
1132                ch.setDisabled(True)
1133            elif ch.objectName() == KEY_SUPTNGSEC_POISSON_RATIO:
1134                ch.setText("0.3")
1135                ch.setDisabled(True)
1136            elif ch.objectName() == KEY_SUPTNGSEC_THERMAL_EXP:
1137                ch.setText("12")
1138                ch.setDisabled(True)
1139            elif ch.objectName() == KEY_SUPTNGSEC_MASS:
1140                ch.setText(str(col_attributes.mass))
```

```
            elif ch.objectName() == KEY_SUPTNGSEC_SEC_AREA:
                ch.setText(str(col_attributes.area))
            elif ch.objectName() == KEY_SUPTNGSEC_MOA_LZ:
                ch.setText(str(col_attributes.mom_inertia_z))
            elif ch.objectName() == KEY_SUPTNGSEC_MOA_LY:
                ch.setText(str(col_attributes.mom_inertia_y))
            elif ch.objectName() == KEY_SUPTNGSEC_ROG_RZ:
                ch.setText(str(col_attributes.rad_of_gy_z))
            elif ch.objectName() == KEY_SUPTNGSEC_ROG_RY:
                ch.setText(str(col_attributes.rad_of_gy_y))
            elif ch.objectName() == KEY_SUPTNGSEC_EM_ZZ:
                ch.setText(str(col_attributes.elast_sec_mod_z))
            elif ch.objectName() == KEY_SUPTNGSEC_EM_ZY:
                ch.setText(str(col_attributes.elast_sec_mod_y))
            elif ch.objectName() == KEY_SUPTNGSEC_PM_ZPZ:
                ch.setText(str(col_attributes.plast_sec_mod_z))
            elif ch.objectName() == KEY_SUPTNGSEC_PM_ZPY:
                ch.setText(str(col_attributes.plast_sec_mod_y))
            elif ch.objectName() == 'pushButton_Add_Column':
                ch.setEnabled(True)
            else:
                pass

        for e in col_list:
            if e.text() != "":
                e.textChanged.connect(lambda:
                ↪  self.new_sectionalprop_Column(col_list))

    def beam_preferences(self, designation, material_grade):
        '''
        @author: Umair
        '''
        if designation == 'Select Section':
            self.ui.clear_tab("Beam")
            return
        beam_attributes = Section(designation, material_grade)
        Section.connect_to_database_update_other_attributes(beam_attributes,
        ↪  "Beams", designation)
        beam_list = []
        for ch in self.ui.tab_Beam.children():
            if ch.objectName() == KEY_SUPTDSEC_DESIGNATION:
                ch.setText(designation)
            elif ch.objectName() == KEY_SUPTDSEC_SOURCE:
                ch.setText(beam_attributes.source)
            elif ch.objectName() == KEY_SUPTDSEC_FU:
                ch.setText(str(beam_attributes.fu))
            elif ch.objectName() == KEY_SUPTDSEC_FY:
                ch.setText(str(beam_attributes.fy))
            elif ch.objectName() == KEY_SUPTDSEC_DEPTH:
                ch.setText(str(beam_attributes.depth))
                beam_list.append(ch)
            elif ch.objectName() == KEY_SUPTDSEC_FLANGE_W:
                ch.setText(str(beam_attributes.flange_width))
                beam_list.append(ch)
            elif ch.objectName() == KEY_SUPTDSEC_FLANGE_T:
                ch.setText(str(beam_attributes.flange_thickness))
```

```
1195                beam_list.append(ch)
1196            elif ch.objectName() == KEY_SUPTDSEC_WEB_T:
1197                ch.setText(str(beam_attributes.web_thickness))
1198                beam_list.append(ch)
1199            elif ch.objectName() == KEY_SUPTDSEC_FLANGE_S:
1200                ch.setText(str(beam_attributes.flange_slope))
1201            elif ch.objectName() == KEY_SUPTDSEC_ROOT_R:
1202                ch.setText(str(beam_attributes.root_radius))
1203            elif ch.objectName() == KEY_SUPTDSEC_TOE_R:
1204                ch.setText(str(beam_attributes.toe_radius))
1205            elif ch.objectName() == KEY_SUPTDSEC_MOD_OF_ELAST:
1206                ch.setText("200")
1207                ch.setDisabled(True)
1208            elif ch.objectName() == KEY_SUPTDSEC_MOD_OF_RIGID:
1209                ch.setText("76.9")
1210                ch.setDisabled(True)
1211            elif ch.objectName() == KEY_SUPTDSEC_POISSON_RATIO:
1212                ch.setText("0.3")
1213                ch.setDisabled(True)
1214            elif ch.objectName() == KEY_SUPTDSEC_THERMAL_EXP:
1215                ch.setText("12")
1216                ch.setDisabled(True)
1217            elif ch.objectName() == KEY_SUPTDSEC_MASS:
1218                ch.setText(str(beam_attributes.mass))
1219            elif ch.objectName() == KEY_SUPTDSEC_SEC_AREA:
1220                ch.setText(str(beam_attributes.area))
1221            elif ch.objectName() == KEY_SUPTDSEC_MOA_LZ:
1222                ch.setText(str(beam_attributes.mom_inertia_z))
1223            elif ch.objectName() == KEY_SUPTDSEC_MOA_LY:
1224                ch.setText(str(beam_attributes.mom_inertia_y))
1225            elif ch.objectName() == KEY_SUPTDSEC_ROG_RZ:
1226                ch.setText(str(beam_attributes.rad_of_gy_z))
1227            elif ch.objectName() == KEY_SUPTDSEC_ROG_RY:
1228                ch.setText(str(beam_attributes.rad_of_gy_y))
1229            elif ch.objectName() == KEY_SUPTDSEC_EM_ZZ:
1230                ch.setText(str(beam_attributes.elast_sec_mod_z))
1231            elif ch.objectName() == KEY_SUPTDSEC_EM_ZY:
1232                ch.setText(str(beam_attributes.elast_sec_mod_y))
1233            elif ch.objectName() == KEY_SUPTDSEC_PM_ZPZ:
1234                ch.setText(str(beam_attributes.plast_sec_mod_z))
1235            elif ch.objectName() == KEY_SUPTDSEC_PM_ZPY:
1236                ch.setText(str(beam_attributes.plast_sec_mod_y))
1237            elif ch.objectName() == 'pushButton_Add_Beam':
1238                ch.setEnabled(True)
1239            else:
1240                pass
1241
1242        for e in beam_list:
1243            if e.text() != "":
1244                e.textChanged.connect(lambda:
                    ↪  self.new_sectionalprop_Beam(beam_list))
1245
1246    def new_sectionalprop_Column(self, col_list):
1247        '''
1248        @author: Umair
1249        '''
```

```
1250
1251            for e in col_list:
1252                if e.text() != "":
1253                    if e.objectName() == KEY_SUPTNGSEC_DEPTH:
1254                        D = float(e.text())
1255                    elif e.objectName() == KEY_SUPTNGSEC_FLANGE_W:
1256                        B = float(e.text())
1257                    elif e.objectName() == KEY_SUPTNGSEC_FLANGE_T:
1258                        t_w = float(e.text())
1259                    elif e.objectName() == KEY_SUPTNGSEC_WEB_T:
1260                        t_f = float(e.text())
1261                    else:
1262                        pass
1263                else:
1264                    return
1265            if col_list:
1266                for c in self.ui.tab_Column.children():
1267                    if c.objectName() == KEY_SUPTNGSEC_MASS:
1268                        c.setText(str(self.sectionalprop.calc_Mass(D, B, t_w,
                            ↪  t_f)))
1269                    elif c.objectName() == KEY_SUPTNGSEC_SEC_AREA:
1270                        c.setText(str(self.sectionalprop.calc_Area(D, B, t_w,
                            ↪  t_f)))
1271                    elif c.objectName() == KEY_SUPTNGSEC_MOA_LZ:
1272                        c.setText(str(self.sectionalprop.calc_MomentOfAreaZ(D, B,
                            ↪  t_w, t_f)))
1273                    elif c.objectName() == KEY_SUPTNGSEC_MOA_LY:
1274                        c.setText(str(self.sectionalprop.calc_MomentOfAreaY(D, B,
                            ↪  t_w, t_f)))
1275                    elif c.objectName() == KEY_SUPTNGSEC_ROG_RZ:
1276                        c.setText(str(self.sectionalprop.calc_RogZ(D, B, t_w,
                            ↪  t_f)))
1277                    elif c.objectName() == KEY_SUPTNGSEC_ROG_RY:
1278                        c.setText(str(self.sectionalprop.calc_RogY(D, B, t_w,
                            ↪  t_f)))
1279                    elif c.objectName() == KEY_SUPTNGSEC_EM_ZZ:
1280                        c.setText(str(self.sectionalprop.calc_ElasticModulusZz(D,
                            ↪  B, t_w, t_f)))
1281                    elif c.objectName() == KEY_SUPTNGSEC_EM_ZY:
1282                        c.setText(str(self.sectionalprop.calc_ElasticModulusZy(D,
                            ↪  B, t_w, t_f)))
1283                    elif c.objectName() == KEY_SUPTNGSEC_PM_ZPZ:
1284                        c.setText(str(self.sectionalprop.calc_PlasticModulusZpz(D,
                            ↪  B, t_w, t_f)))
1285                    elif c.objectName() == KEY_SUPTNGSEC_PM_ZPY:
1286                        c.setText(str(self.sectionalprop.calc_PlasticModulusZpy(D,
                            ↪  B, t_w, t_f)))
1287                    elif c.objectName() == 'pushButton_Add_Column':
1288                        c.setEnabled(True)
1289                    else:
1290                        pass
1291
1292        def new_sectionalprop_Beam(self, beam_list):
1293            '''
1294            @author: Umair
1295            '''
```

```
1296
1297            for e in beam_list:
1298                if e.text() != "":
1299                    if e.objectName() == KEY_SUPTDSEC_DEPTH:
1300                        D = float(e.text())
1301                    elif e.objectName() == KEY_SUPTDSEC_FLANGE_W:
1302                        B = float(e.text())
1303                    elif e.objectName() == KEY_SUPTDSEC_FLANGE_T:
1304                        t_w = float(e.text())
1305                    elif e.objectName() == KEY_SUPTDSEC_WEB_T:
1306                        t_f = float(e.text())
1307                    else:
1308                        pass
1309                else:
1310                    return
1311            if beam_list:
1312                for c in self.ui.tab_Beam.children():
1313                    if c.objectName() == KEY_SUPTDSEC_MASS:
1314                        c.setText(str(self.sectionalprop.calc_Mass(D, B, t_w,
                            ↪  t_f)))
1315                    elif c.objectName() == KEY_SUPTDSEC_SEC_AREA:
1316                        c.setText(str(self.sectionalprop.calc_Area(D, B, t_w,
                            ↪  t_f)))
1317                    elif c.objectName() == KEY_SUPTDSEC_MOA_LZ:
1318                        c.setText(str(self.sectionalprop.calc_MomentOfAreaZ(D, B,
                            ↪  t_w, t_f)))
1319                    elif c.objectName() == KEY_SUPTDSEC_MOA_LY:
1320                        c.setText(str(self.sectionalprop.calc_MomentOfAreaY(D, B,
                            ↪  t_w, t_f)))
1321                    elif c.objectName() == KEY_SUPTDSEC_ROG_RZ:
1322                        c.setText(str(self.sectionalprop.calc_RogZ(D, B, t_w,
                            ↪  t_f)))
1323                    elif c.objectName() == KEY_SUPTDSEC_ROG_RY:
1324                        c.setText(str(self.sectionalprop.calc_RogY(D, B, t_w,
                            ↪  t_f)))
1325                    elif c.objectName() == KEY_SUPTDSEC_EM_ZZ:
1326                        c.setText(str(self.sectionalprop.calc_ElasticModulusZz(D,
                            ↪  B, t_w, t_f)))
1327                    elif c.objectName() == KEY_SUPTDSEC_EM_ZY:
1328                        c.setText(str(self.sectionalprop.calc_ElasticModulusZy(D,
                            ↪  B, t_w, t_f)))
1329                    elif c.objectName() == KEY_SUPTDSEC_PM_ZPZ:
1330                        c.setText(str(self.sectionalprop.calc_PlasticModulusZpz(D,
                            ↪  B, t_w, t_f)))
1331                    elif c.objectName() == KEY_SUPTDSEC_PM_ZPY:
1332                        c.setText(str(self.sectionalprop.calc_PlasticModulusZpy(D,
                            ↪  B, t_w, t_f)))
1333                    elif c.objectName() == 'pushButton_Add_Beam':
1334                        c.setEnabled(True)
1335                    else:
```

Appendix H

Code for Output Dock

```
739         out_list = main.output_values(main, False)
740         _translate = QtCore.QCoreApplication.translate
741 
742         i = 0
743         for option in out_list:
744             lable = option[1]
745             type = option[2]
746             if type not in [TYPE_TITLE, TYPE_IMAGE, TYPE_MODULE]:
747                 l = QtWidgets.QLabel(self.dockWidgetContents_out)
748                 l.setGeometry(QtCore.QRect(6, 10 + i, 120, 25))
749                 font = QtGui.QFont()
750                 font.setPointSize(11)
751                 font.setBold(False)
752                 font.setWeight(50)
753                 l.setFont(font)
754                 l.setObjectName(option[0] + "_label")
755                 l.setText(_translate("MainWindow", "<html><head/><body><p>" +
                    ↪  lable + "</p></body></html>"))
756 
757             if type == TYPE_TEXTBOX:
758                 r = QtWidgets.QLineEdit(self.dockWidgetContents_out)
759                 r.setGeometry(QtCore.QRect(150, 10 + i, 160, 27))
760                 font = QtGui.QFont()
761                 font.setPointSize(11)
762                 font.setBold(False)
763                 font.setWeight(50)
764                 r.setFont(font)
765                 r.setObjectName(option[0])
766 
767             if type == TYPE_OUT_BUTTON:
768                 v = option[3]
769                 b = QtWidgets.QPushButton(self.dockWidgetContents_out)
770                 b.setGeometry(QtCore.QRect(150, 10 + i, 160, 27))
771                 font = QtGui.QFont()
772                 font.setPointSize(11)
773                 font.setBold(False)
774                 font.setWeight(50)
775                 b.setFont(font)
776                 b.setObjectName(option[0])
777                 b.setText(v[0])
```

```
778                 b.setDisabled(True)
779                 b.clicked.connect(lambda: self.output_button_dialog(main, v))
780 
781             if type == TYPE_TITLE:
782                 q = QtWidgets.QLabel(self.dockWidgetContents_out)
783                 q.setGeometry(QtCore.QRect(3, 10 + i, 201, 25))
784                 font = QtGui.QFont()
785                 q.setFont(font)
786                 q.setObjectName("_title")
787                 q.setText(_translate("MainWindow",
788                                      "<html><head/><body><p><span style=\"
                                     ↪  font-weight:600;\">" + lable +
                                     ↪  "</span></p></body></html>"))
789             i = i + 30
790 
791         self.outputDock.setWidget(self.dockWidgetContents_out)
```

Appendix I

Code for Reload

```
1420
1421    def refresh_sections(self, prev, section):
1422
1423        connectivity = self.dockWidgetContents.findChild(QtWidgets.QWidget,
        ↪  KEY_CONN)
1424        supporting_section =
        ↪  self.dockWidgetContents.findChild(QtWidgets.QWidget, KEY_SUPTNGSEC)
1425        supported_section =
        ↪  self.dockWidgetContents.findChild(QtWidgets.QWidget, KEY_SUPTDSEC)
1426        Columns = connectdb("Columns")
1427        Beams = connectdb("Beams")
1428        red_list_set = set(red_list_function())
1429
1430        if section == "Supporting":
1431            supporting_section.clear()
1432            if connectivity.currentText() in VALUES_CONN_1:
1433                for item in Columns:
1434                    supporting_section.addItem(item)
1435                current_list_set = set(Columns)
1436                current_red_list =
                ↪  list(current_list_set.intersection(red_list_set))
1437                for value in current_red_list:
1438                    indx = Columns.index(str(value))
1439                    supporting_section.setItemData(indx, QBrush(QColor("red")),
                    ↪  Qt.TextColorRole)
1440
1441            elif connectivity.currentText() in VALUES_CONN_2:
1442                for item in Beams:
1443                    supporting_section.addItem(item)
1444                current_list_set = set(Beams)
1445                current_red_list =
                ↪  list(current_list_set.intersection(red_list_set))
1446                for value in current_red_list:
1447                    indx = Beams.index(str(value))
1448                    supporting_section.setItemData(indx, QBrush(QColor("red")),
                    ↪  Qt.TextColorRole)
1449            text = self.designPrefDialog.findChild(QtWidgets.QWidget,
            ↪  KEY_SUPTNGSEC_DESIGNATION).text()
1450            text_index = supporting_section.findText(text,
            ↪  QtCore.Qt.MatchFixedString)
1451            if text_index:
```

```
                supporting_section.setCurrentIndex(text_index)
            else:
                supporting_section.setCurrentIndex(prev)

        if section == "Supported":
            supported_section.clear()

            for item in Beams:
                supported_section.addItem(item)
            current_list_set = set(Beams)
            current_red_list =
            ↪  list(current_list_set.intersection(red_list_set))
            for value in current_red_list:
                indx = Beams.index(str(value))
                supported_section.setItemData(indx, QBrush(QColor("red")),
                ↪  Qt.TextColorRole)
            text = self.designPrefDialog.findChild(QtWidgets.QWidget,
            ↪  KEY_SUPTDSEC_DESIGNATION).text()
            text_index = supported_section.findText(text,
            ↪  QtCore.Qt.MatchFixedString)
            if text_index:
                supported_section.setCurrentIndex(text_index)
            else:
                supported_section.setCurrentIndex(prev)

```

Appendix J

Code for Dialog box in Output-dock

```
1386
1387    def output_button_dialog(self, main, list):
1388        dialog = QtWidgets.QDialog()
1389        dialog.resize(350, 170)
1390        dialog.setFixedSize(dialog.size())
1391        dialog.setObjectName("Dialog")
1392        dialog.setWindowTitle(list[0])
1393        i = 0
1394        for option in list[1](main, main.design_status):
1395            lable = option[1]
1396            type = option[2]
1397            _translate = QtCore.QCoreApplication.translate
1398            if type not in [TYPE_TITLE, TYPE_IMAGE, TYPE_MODULE]:
1399                l = QtWidgets.QLabel(dialog)
1400                l.setGeometry(QtCore.QRect(10, 10 + i, 120, 25))
1401                font = QtGui.QFont()
1402                font.setPointSize(9)
1403                font.setBold(False)
1404                font.setWeight(50)
1405                l.setFont(font)
1406                l.setObjectName(option[0] + "_label")
1407                l.setText(_translate("MainWindow", "<html><head/><body><p>" +
                    ↪  lable + "</p></body></html>"))
1408            if type == TYPE_TEXTBOX:
1409                r = QtWidgets.QLineEdit(dialog)
1410                r.setGeometry(QtCore.QRect(160, 10 + i, 160, 27))
1411                font = QtGui.QFont()
1412                font.setPointSize(11)
1413                font.setBold(False)
1414                font.setWeight(50)
1415                r.setFont(font)
1416                r.setObjectName(option[0])
1417                r.setText(str(option[3]))
1418            i = i + 30
1419        dialog.exec()
1420
```