```
"""The controller module for the minor misdemeanor dialog - it is not limited
to minor misdemeanors, but does not contain functions to account for jail time.
Loads all charges - including non-minor-misdemeanors from a database."""
import pathlib
from datetime import date, timedelta
from loguru import logger

from PyQt5 import QtCore
from PyQt5.QtCore import QDate
from PyQt5.QtWidgets import QLabel, QPushButton, QMessageBox, QComboBox, QLineEdit
from PyQt5.QtSql import QSqlDatabase, QSqlQuery

from views.custom_widgets import (
    PleaComboBox,
    FindingComboBox,
    FineLineEdit,
    FineSuspendedLineEdit,
    DeleteButton,
    AmendButton,
)
from views.minor_misdemeanor_dialog_ui import Ui_MinorMisdemeanorDialog
from models.template_types import TEMPLATE_DICT
from models.case_information import (
    CaseInformation,
    CriminalCharge,
    AmendOffenseDetails,
    LicenseSuspensionTerms,
    CommunityControlTerms,
    CommunityServiceTerms,
    OtherConditionsTerms,
)
from models.messages import TURNS_AT_INTERSECTIONS as TURNS_WARNING
from controllers.criminal_dialogs import (
    BaseCriminalDialog,
    AddConditionsDialog,
    AmendOffenseDialog,
)
from resources.db.DatabaseCreation import create_offense_list, create_statute_list


PATH = str(pathlib.Path().absolute())
TEMPLATE_PATH = PATH + "\\resources\\templates\\"
SAVE_PATH = PATH + "\\resources\\saved\\"
DB_PATH = PATH + "\\resources\\db\\"
CHARGES_DATABASE = DB_PATH + "\\charges.sqlite"

```

```
47
48  @logger.catch
49  def create_database_connections():
50      """The databases for the application are created upon import of the module, which is done
51      on application startup. The connections to the databases are created, but the opening and
52      closing of the databases is handled by the appropriate class dialog."""
53      offense_database_connection = QSqlDatabase.addDatabase("QSQLITE", "offenses")
54      offense_database_connection.setDatabaseName(CHARGES_DATABASE)
55      statute_database_connection = QSqlDatabase.addDatabase("QSQLITE", "statutes")
56      statute_database_connection.setDatabaseName(CHARGES_DATABASE)
57      return offense_database_connection, statute_database_connection
58
59
60  @logger.catch
61  def open_databases():
62      """
63      https://www.tutorialspoint.com/pyqt/pyqt_database_handling.htm
64      https://doc.qt.io/qtforpython/overviews/sql-connecting.html
65      NOTE: If running create_psql_table.py to update database, must delete
66      the old charges.sqlite file to insure it is updated.
67      """
68      database_offenses.open()
69      database_statutes.open()
70
71
72  @logger.catch
73  def close_databases():
74      """Closes any databases that were opened at the start of the dialog."""
75      database_offenses.close()
76      database_offenses.removeDatabase(CHARGES_DATABASE)
77      database_statutes.close()
78      database_statutes.removeDatabase(CHARGES_DATABASE)
79
80
81  class MinorMisdemeanorDialog(BaseCriminalDialog, Ui_MinorMisdemeanorDialog):
82      """The dialog inherits from the BaseCriminalDialog (controller) and the
83      Ui_MinorMisdemeanorDialog (view).
84
85      This dialog is used when there will not be any jail time imposed. It does
86      not inherently limit cases to minor misdemeanors or unclassified
87      misdemeanors, however, it does not include fields to enter jail time."""
88
89      @logger.catch
90      def __init__(self, judicial_officer, parent=None):
91          open_databases()
92          super().__init__(parent)
```

```
93              self.case_information = CaseInformation(judicial_officer)
94              self.modify_view()
95              self.connect_signals_to_slots()
96              self.template = TEMPLATE_DICT.get(self.case_information.judicial_officer.last_name)
97              self.delete_button_list = []
98              self.amend_button_list = []
99
100         @logger.catch
101         def modify_view(self):
102             """The modify view method updates the view that is created on init from the
103             Ui_MinorMisdemeanorDialog. Place items in this method that can't be added
104             directly in QtDesigner so that they don't need to be changed in the view file
105             each time pyuic5 is run."""
106             statute_list = create_statute_list()
107             self.statute_choice_box.addItems(statute_list)
108             self.offense_choice_box.addItems(create_offense_list())
109             self.plea_trial_date.setDate(QtCore.QDate.currentDate())
110             self.balance_due_date.setDate(QtCore.QDate.currentDate())
111             self.statute_choice_box.setCurrentText("")
112             self.offense_choice_box.setCurrentText("")
113
114         @logger.catch
115         def connect_signals_to_slots(self):
116             """The method connects any signals to slots. Generally, connecting with
117             pressed is preferred to clicked because a clicked event sends a bool
118             argument to the function. However, clicked is used in some instances
119             because it is a press and release of a button. Using pressed sometimes
120             caused an event to be triggered twice."""
121             self.cancel_Button.pressed.connect(self.close_event)
122             self.clear_fields_case_Button.pressed.connect(self.clear_case_information_fields)
123             self.create_entry_Button.pressed.connect(self.create_entry_process)
124             self.add_conditions_Button.pressed.connect(self.start_add_conditions_dialog)
125             self.add_charge_Button.clicked.connect(self.add_charge_process)
126             self.clear_fields_charge_Button.pressed.connect(self.clear_charge_fields)
127             self.statute_choice_box.currentTextChanged.connect(self.set_offense)
128             self.offense_choice_box.currentTextChanged.connect(self.set_statute)
129             self.fra_in_file_box.currentTextChanged.connect(self.set_fra_in_file)
130             self.fra_in_court_box.currentTextChanged.connect(self.set_fra_in_court)
131             self.ability_to_pay_box.currentTextChanged.connect(self.set_pay_date)
132             self.guilty_all_Button.pressed.connect(self.guilty_all_plea_and_findings)
133             self.no_contest_all_Button.pressed.connect(self.no_contest_all_plea_and_findings)
134             self.costs_and_fines_Button.clicked.connect(self.show_costs_and_fines)
135
136         def clear_case_information_fields(self):
137             self.defendant_first_name_lineEdit.clear()
138             self.defendant_last_name_lineEdit.clear()
```

```
139             self.case_number_lineEdit.clear()
140             self.defendant_first_name_lineEdit.setFocus()
141
142         def clear_charge_fields(self):
143             """Clears the fields that are used for adding a charge. The
144             statute_choice_box and offense_choice_box use the clearEditText
145             method because those boxes are editable."""
146             self.statute_choice_box.clearEditText()
147             self.offense_choice_box.clearEditText()
148
149         @logger.catch
150         def start_amend_offense_dialog(self):
151             """Opens the amend offense dialog as a modal window. The
152             case_information is passed to the dialog class in order to populate
153             the case information banner."""
154             self.update_case_information()
155             AmendOffenseDialog(self.case_information).exec()
156
157         @logger.catch
158         def start_add_conditions_dialog(self):
159             """Opens the add conditions dialog as a modal window. It passes the
160             instance of the MinorMisdemeanorDialog class (self) as an argument
161             so that the AddConditionsDialog can access all data from the
162             MinorMisdemeanorDialog when working in the AddConditionsDialog."""
163             AddConditionsDialog(self).exec()
164
165         @logger.catch
166         def close_event(self):
167             """Place any cleanup items (i.e. close_databases) here that should be
168             called when the entry is created and the dialog closed."""
169             close_databases()
170             self.close_window()
171
172         @logger.catch
173         def add_charge(self):
174             """The add_charge_process, from which this is called, creates a criminal
175             charge object and adds the data in the view to the object. The criminal
176             charge (offense, statute, degree and type) is then added to the case
177             information model (by appending the charge object to the criminal
178             charges list).
179
180             The offense, statute and degree are added to the view by the method
181             add_charge_to_view, not this method. This method is triggered on
182             clicked() of the Add Charge button."""
183             self.criminal_charge.offense = self.offense_choice_box.currentText()
184             self.criminal_charge.statute = self.statute_choice_box.currentText()
```

```
185             self.criminal_charge.degree = self.degree_choice_box.currentText()
186             self.criminal_charge.type = self.set_offense_type()
187             self.case_information.add_charge_to_list(self.criminal_charge)
188             self.add_charge_to_view()
189             self.statute_choice_box.setFocus()
190
191         @logger.catch
192         def add_charge_to_view(self):
193             """Adds the charge that was added through add_charge method to the
194             view/GUI. The first row=0 because of python zero-based indexing. The
195             column is set at one more than the current number of columns because
196             it is the column to which the charge will be added.
197
198             :added_charge_index: - The added charge index is one less than the
199             total charges in charges_list because of zero-based indexing. Thus, if
200             there is one charge, the index of the charge to be added to the
201             charge_dict from the charges_list is 0.
202
203             The python builtin vars function returns the __dict__ attribute of
204             the object.
205
206             The self.criminal_charge.offense added as a parameter for FineLineEdit
207             is the current one added when "Add Charge" is pressed.
208
209             TODO: Refactor so that there isn't a need for a if branch to skip the
210             attribute for charge type."""
211             row = 0
212             column = self.charges_gridLayout.columnCount() + 1
213             added_charge_index = len(self.case_information.charges_list) - 1
214             charge = vars(self.case_information.charges_list[added_charge_index])
215             for value in charge.values():
216                 if value is not None:
217                     if value in ["Moving Traffic", "Non-moving Traffic", "Criminal"]:
218                         break
219                     self.charges_gridLayout.addWidget(QLabel(value), row, column)
220                     row += 1
221             self.charges_gridLayout.addWidget(PleaComboBox(), row, column)
222             row +=1
223             self.charges_gridLayout.addWidget(FindingComboBox(), row, column)
224             row +=1
225             self.charges_gridLayout.addWidget(FineLineEdit(self.criminal_charge.offense), row,
226     column)
227             row +=1
228             self.charges_gridLayout.addWidget(FineSuspendedLineEdit(), row, column)
229             row +=1
230             self.add_delete_button_to_view(row, column)
```

```
231             row +=1
232             self.add_amend_button_to_view(row, column)
233
234         def add_delete_button_to_view(self, row, column):
235             delete_button = DeleteButton()
236             self.delete_button_list.append(delete_button)
237             delete_button.pressed.connect(self.delete_charge)
238             self.charges_gridLayout.addWidget(delete_button, row, column)
239
240         def add_amend_button_to_view(self, row, column):
241             amend_button = AmendButton()
242             self.amend_button_list.append(amend_button)
243             amend_button.pressed.connect(self.start_amend_offense_dialog)
244             self.charges_gridLayout.addWidget(amend_button, row, column)
245
246         @logger.catch
247         def delete_charge(self):
248             """Deletes the offense from the case_information.charges list. Then
249             decrements the total charges by one so that other functions using the
250             total charges for indexing are correct."""
251             index = self.delete_button_list.index(self.sender())
252             del self.case_information.charges_list[index]
253             del self.delete_button_list[index]
254             self.delete_charge_from_view()
255             self.statute_choice_box.setFocus()
256
257         @logger.catch
258         def delete_charge_from_view(self):
259             """Uses the delete_button that is indexed to the column to delete the
260             QLabels for the charge."""
261             index = self.charges_gridLayout.indexOf(self.sender())
262             column = self.charges_gridLayout.getItemPosition(index)[1]
263             for row in range(self.charges_gridLayout.rowCount()):
264                 layout_item = self.charges_gridLayout.itemAtPosition(row, column)
265                 if layout_item is not None:
266                     layout_item.widget().deleteLater()
267                     self.charges_gridLayout.removeItem(layout_item)
268
269         @logger.catch
270         def update_case_information(self):
271             """The method calls functions to update the case information model
272             with the data for the case that is in the fields on the view. This does
273             not update the model
274             with information in the charge fields (offense, statute, plea, etc.)
275             the charge information is transferred to the model upon press of the
276             add charge button.
```

```
277
278             Fields that are updated upon pressed() of createEntryButton = case
279             number, first name, last name, ability to pay time, balance due date,
280             date of plea/trial,operator license number, date of birth, FRA (proof
281             of insurance) in complaint, FRA in court."""
282             self.update_party_information()
283             self.update_costs_and_fines_information()
284             self.add_dispositions_and_fines()
285             self.check_add_conditions()
286             self.calculate_costs_and_fines()
287
288         @logger.catch
289         def update_party_information(self):
290             self.case_information.case_number = self.case_number_lineEdit.text()
291             self.case_information.defendant.first_name = self.defendant_first_name_lineEdit.text()
292             self.case_information.defendant.last_name = self.defendant_last_name_lineEdit.text()
293             self.case_information.plea_trial_date = self.plea_trial_date.date().toString("MMMM dd,
294     yyyy")
295
296         @logger.catch
297         def update_costs_and_fines_information(self):
298             self.case_information.court_costs_ordered = self.court_costs_box.currentText()
299             self.case_information.ability_to_pay_time = self.ability_to_pay_box.currentText()
300             self.case_information.balance_due_date =
301     self.balance_due_date.date().toString("MMMM dd, yyyy")
302
303         @logger.catch
304         def add_dispositions_and_fines(self):
305             """Row 3 - plea, 4 - finding, 5 - fine, 6 fine-suspended.
306             Columns start at 0 for labels and 2 for first entry then 4 etc.
307
308             Column count increases by 2 instead of one due to grid adding two
309             columns when a charge is added (odd numbered column is empty)."""
310             column = 2
311             try:
312                 for index in range(len(self.case_information.charges_list)):
313                     self.case_information.charges_list[index].plea =
314     self.charges_gridLayout.itemAtPosition(3,column).widget().currentText()
315                     self.case_information.charges_list[index].finding =
316     self.charges_gridLayout.itemAtPosition(4,column).widget().currentText()
317                     self.case_information.charges_list[index].fines_amount =
318     self.charges_gridLayout.itemAtPosition(5,column).widget().text()
319                     if self.charges_gridLayout.itemAtPosition(6,column).widget().text() == "":
320                         self.case_information.charges_list[index].fines_suspended = "0"
321                     else:
322                         self.case_information.charges_list[index].fines_suspended =
```

```
323     self.charges_gridLayout.itemAtPosition(6,column).widget().text()
324                         index +=1
325                         column +=2
326             except AttributeError:
327                 print("Attribute error allowed to pass for lack of widget")
328 
329         @logger.catch
330         def check_add_conditions(self):
331             """Checks to see what conditions in the Add Conditions box are checked and then
332             transfers the information from the conditions to case_information model if the
333             box is checked.
334 
335             TODO:
336             in future refactor this to have it loop through the different conditions so code
337             doesn't need to be added each time a condition is added."""
338             if self.license_suspension_checkBox.isChecked():
339                 self.case_information.license_suspension_details.license_suspension_ordered = (
340                     True
341                 )
342             if self.community_control_checkBox.isChecked():
343                 self.case_information.community_control_terms.community_control_required = (
344                     True
345                 )
346             if self.community_service_checkBox.isChecked():
347                 self.case_information.community_service_terms.community_service_ordered = (
348                     True
349                 )
350             if self.other_conditions_checkBox.isChecked():
351                 self.case_information.other_conditions_details.other_conditions_ordered = (
352                     True
353                 )
354 
355         @logger.catch
356         def calculate_costs_and_fines(self):
357             self.case_information.court_costs = 0
358             if self.court_costs_box.currentText() == "Yes":
359                 for index, charge in enumerate(self.case_information.charges_list):
360                     if self.case_information.court_costs == 124:
361                         break
362                     else:
363                         if charge.type == "Moving Traffic":
364                             if self.case_information.court_costs < 124:
365                                 self.case_information.court_costs = 124
366                         elif charge.type == "Criminal":
367                             if self.case_information.court_costs < 114:
368                                 self.case_information.court_costs = 114
```

```
                        elif charge.type == "Non-moving Traffic":
                            if self.case_information.court_costs < 95:
                                self.case_information.court_costs = 95
        total_fines = 0
        try:
            for index, charge in enumerate(self.case_information.charges_list):
                if charge.fines_amount == '':
                    charge.fines_amount = 0
                total_fines = total_fines + int(charge.fines_amount)
            self.case_information.total_fines = total_fines
            total_fines_suspended = 0
            for index, charge in enumerate(self.case_information.charges_list):
                if charge.fines_suspended == '':
                    charge.fines_suspended = 0
                total_fines_suspended = total_fines_suspended + int(charge.fines_suspended)
            self.case_information.total_fines_suspended = total_fines_suspended
        except TypeError:
            print("A type error was allowed to pass - this is because of deleted charge.")

    def show_costs_and_fines(self, bool):
        """The bool is the toggle from the clicked() of the button pressed. No
        action is taken with respect to it."""
        self.update_case_information()
        message = QMessageBox()
        message.setIcon(QMessageBox.Information)
        message.setWindowTitle("Total Costs and Fines")
        message.setInformativeText("Costs: $" + str(self.case_information.court_costs) +\
            "\nFines: $" + str(self.case_information.total_fines) +\
            "\nFines Suspended: $" + str(self.case_information.total_fines_suspended) +\
            "\n\n*Does not include possible bond forfeiture or other costs \n that may be assesed
as a result of prior actions in case. ")
        total_fines_and_costs = (self.case_information.court_costs +\
            self.case_information.total_fines) - self.case_information.total_fines_suspended
        message.setText("Total Costs and Fines Due By Due Date: $" +
str(total_fines_and_costs))
        message.setStandardButtons(QMessageBox.Ok)
        message.exec_()

    def guilty_all_plea_and_findings(self):
        """Sets the plea and findings boxes to guilty for all charges currently
        in the charges_gridLayout."""
        for column in range(self.charges_gridLayout.columnCount()):
            try:
                if isinstance(self.charges_gridLayout.itemAtPosition(3, column).widget(),
PleaComboBox):

```

```
415     self.charges_gridLayout.itemAtPosition(3,column).widget().setCurrentText("Guilty")
416
417     self.charges_gridLayout.itemAtPosition(4,column).widget().setCurrentText("Guilty")
418                             column +=1
419                 except AttributeError:
420                     pass
421             self.set_cursor_to_FineLineEdit()
422
423         def no_contest_all_plea_and_findings(self):
424             """Sets the plea box to no contest and findings boxes to guilty for all
425             charges currently in the charges_gridLayout."""
426             for column in range(self.charges_gridLayout.columnCount()):
427                 try:
428                     if isinstance(self.charges_gridLayout.itemAtPosition(3, column).widget(),
429     PleaComboBox):
430
431     self.charges_gridLayout.itemAtPosition(3,column).widget().setCurrentText("No Contest")
432
433     self.charges_gridLayout.itemAtPosition(4,column).widget().setCurrentText("Guilty")
434                             column +=1
435                 except AttributeError:
436                     pass
437             self.set_cursor_to_FineLineEdit()
438
439         def set_cursor_to_FineLineEdit(self):
440             for column in range(self.charges_gridLayout.columnCount()):
441                 try:
442                     if isinstance(self.charges_gridLayout.itemAtPosition(5, column).widget(),
443     FineLineEdit):
444                             self.charges_gridLayout.itemAtPosition(5, column).widget().setFocus()
445                             break
446                 except AttributeError:
447                     pass
448
449         @logger.catch
450         def set_fra_in_file(self, current_text):
451             """Sets the FRA (proof of insurance) to true if the view indicates 'yes'
452             that the FRA was shown in the complaint of file."""
453             if current_text == "Yes":
454                 self.case_information.fra_in_file = True
455                 self.fra_in_court_box.setCurrentText("No")
456             elif current_text == "No":
457                 self.case_information.fra_in_file = False
458             else:
459                 self.case_information.fra_in_file = None
460
```

```
461         @logger.catch
462         def set_fra_in_court(self, current_text):
463             """Sets the FRA (proof of insurance) to true if the view indicates 'yes'
464             that the FRA was shown in court."""
465             if current_text == "Yes":
466                 self.case_information.fra_in_court = True
467             elif current_text == "No":
468                 self.case_information.fra_in_court = False
469             else:
470                 self.case_information.fra_in_court = None
471 
472         def set_offense_type(self):
473             key = self.statute_choice_box.currentText()
474             if self.freeform_entry_checkBox.isChecked():
475                 return None
476             query = QSqlQuery(database_statutes)
477             query.prepare("SELECT * FROM charges WHERE " "statute LIKE '%' || :key || '%'")
478             query.bindValue(":key", key)
479             query.exec()
480             while query.next():
481                 statute = query.value(2)
482                 offense_type = query.value(4)
483                 if statute == key:
484                     return offense_type
485 
486         @logger.catch
487         def set_statute(self, key):
488             """This method queries based on the offense and then sets the statute
489             and degree based on the offense in the database.
490 
491             :key: is the string that is passed by the function each time the field
492             is changed on the view. This is the offense."""
493             if self.freeform_entry_checkBox.isChecked():
494                 return None
495             query = QSqlQuery(database_offenses)
496             query.prepare("SELECT * FROM charges WHERE " "offense LIKE '%' || :key || '%'")
497             query.bindValue(":key", key)
498             query.exec()
499             while query.next():
500                 offense = query.value(1)
501                 statute = query.value(2)
502                 degree = query.value(3)
503                 if offense == key:
504                     self.statute_choice_box.setCurrentText(statute)
505                     self.degree_choice_box.setCurrentText(degree)
506                     break
```

```
507
508         @logger.catch
509         def set_offense(self, key):
510             """This method queries based on the statute and then sets the offense
511             and degree based on the statute in the database.
512
513             :key: is the string that is passed by the function each time the field
514             is changed on the view. This is the statute."""
515             if self.freeform_entry_checkBox.isChecked():
516                 return None
517             query = QSqlQuery(database_statutes)
518             query.prepare("SELECT * FROM charges WHERE " "statute LIKE '%' || :key || '%'")
519             query.bindValue(":key", key)
520             query.exec()
521             while query.next():
522                 offense = query.value(1)
523                 statute = query.value(2)
524                 degree = query.value(3)
525                 if statute == key:
526                     self.offense_choice_box.setCurrentText(offense)
527                     self.degree_choice_box.setCurrentText(degree)
528                     break
529
530         @logger.catch
531         def set_pay_date(self, time_to_pay_text):
532             """Sets the balance of fines and costs to a future date (or today)
533             depending on the selection of ability_to_pay_box. The inner function
534             will move the actual date to the next tuesday per court procedure for
535             show cause hearings being on Tuesday. Would need to be modified if the
536             policy changed."""
537             days_to_add = self.pay_date_dict[time_to_pay_text]
538             future_date = date.today() + timedelta(days_to_add)
539             today = date.today()
540
541             def next_tuesday(future_date, weekday=1):
542                 """This function returns the number of days to add to today to set
543                 the payment due date out to the Tuesday after the number of days
544                 set in the set_pay_date function. The default of 1 for weekday is
545                 what sets it to a Tuesday. If it is 0 it would be Monday, 3 would
546                 be Wednesday, etc."""
547                 days_ahead = weekday - future_date.weekday()
548                 if days_ahead <= 0:   # Target day already happened this week
549                     days_ahead += 7
550                 return future_date + timedelta(days_ahead)
551
552             future_date = next_tuesday(future_date, 1)
```

```
            total_days_to_add = (future_date - today).days
            self.balance_due_date.setDate(QDate.currentDate().addDays(total_days_to_add))


if __name__ == "__main__":
    print("MMD ran directly")
else:
    print("MMD ran when imported")
    database_offenses, database_statutes = create_database_connections()
```