Homework 4

Software Engineering Principle

Software Engineering Program,

Department of Computer Engineering,

School of Engineering, KMITL

67011352 Theepakorn Phayonrat

Features

Gantt Chart

Used for planning project plan and tasks duration or deadline.

Class Diagram

Used for designing classes in the projects.

Interaction Diagram

Used for designing how classes interact each others in the projects.

Markdown Renderer for the task assignment page

How it works:

- As mentioned earlier, we can use markdown to express the task, $\therefore$ we need a markdown renderer.

Implementation Approach:

- Use QEngineWebView Module in PyQt.

VS-Code Extension (OPTIONAL)

`TODO` extension in VS-Code with better description for the task and with team member(s) assigned to that task.
How it works:

- If you have comment with `TODO` in the front, you can add description of the task in a different entry and also in a markdown file.
- If you want to add a person in charge for that task (OPTIONAL), you can use `@TEMP`, where `TEMP` can be either role or team member names.
- After saved, you can access the `TODO` description as you hover and click to inspect task in the comment.

Implementation Approach:

- Scan through the file looking for comment with `TODO` in the front then keep the entry into the DB.
- We can edit the `TODO` description inside a external markdown file.

Page included in this homework

- **Task Assignment Page**: Page to edit `TODO` for task assignments with markdown supported for better view. User can choose whether to edit in the manual mode or external text editor and save file because it can also fetch from real `.md` files in the real program.

Code:

TaskAssignmentPageCode.py

```python
import sys
from PySide6.QtWidgets import *
from PySide6.QtCore import *
from PySide6.QtGui import *
from PySide6.QtWebEngineWidgets import *
import markdown as md

from TaskAssignmentPageUI import Ui_Form

class TaskAssignmentPage(QWidget):
    def __init__(self) -> None:
        QWidget.__init__(self, None)
        self.ui = Ui_Form()
        self.ui.setupUi(self)
        self.sync_preview = QWebEngineView()
        self.manual_preview = QWebEngineView()
        self.ui.web_engine_hbox.addWidget(self.sync_preview,1)
        self.ui.preview_vbox_manual.addWidget(self.manual_preview, ⌋
          ↪  1)
        self.current_mode_text = "Manual"
        self.text = "Type Here..."
        self.ui.editor.setPlaceholderText(self.text)
        self.ui.toggle_btn.clicked.connect(self.toggle_mode)
        self.ui.manual_convert_btn.clicked.connect(self.convert)
        self.ui.sync_convert_btn.clicked.connect(self.convert)
        self.ui.current_mode_label.setText(f"Mode:
          ↪  {self.current_mode_text}")
        self.ui.manual_save_btn.clicked.connect(self.save)
        self.ui.sync_save_btn.clicked.connect(self.save)
        self.ui.manual_load_btn.clicked.connect(self.load)
        self.ui.sync_load_btn.clicked.connect(self.load)
        logo_png = QPixmap("images/logo.png")
        self.ui.logo.setPixmap(logo_png.scaledToHeight(36,
          ↪  Qt.TransformationMode.SmoothTransformation))

    def toggle_mode(self) -> None:
        if self.current_mode_text == "Sync":
            self.current_mode_text = "Manual"
            self.ui.mode_tab.setCurrentIndex(0)
            self.ui.current_mode_label.setText(f"Mode:
              ↪  {self.current_mode_text}")
        else:
            self.current_mode_text = "Sync"
            self.ui.mode_tab.setCurrentIndex(1)
            self.ui.current_mode_label.setText(f"Mode:
              ↪  {self.current_mode_text}")

    def convert(self) -> None:
        self.text = self.ui.editor.toPlainText()
        self.sync_preview.setHtml(md.markdown(self.text))
        self.manual_preview.setHtml(md.markdown(self.text))


    def load(self) -> None:
        file_path, _ = QFileDialog.getOpenFileName(self,
            "Select a Markdown File",
            "",
            "Markdown Files (*.md)"
        )
        print(file_path)
        if not file_path:
            return None
        try:
            with open(file_path, "r+") as file:
                self.text = file.read()
                print(self.text)
                self.ui.editor.setPlainText(self.text)
        except Exception as e:
            d = QDialog(None)
            vbox = QVBoxLayout()
            label = QLabel()
            label.setText(str(e))
            vbox.addWidget(label)
            d.setLayout(vbox)


    def save(self) -> None:
        file_path, _ = QFileDialog.getSaveFileName(self,
                "Select a Markdown File",
                "",
                "Markdown Files (*.md)"
            )
        if not file_path:
            return None
        try:
            with open(file_path, "w+") as file:
                file.write(self.text)
        except Exception as e:
            d = QDialog(None)
            vbox = QVBoxLayout()
            label = QLabel()
            label.setText(str(e))
            vbox.addWidget(label)
            d.setLayout(vbox)


def main() -> None:
    app = QApplication(sys.argv)
    w = TaskAssignmentPage()
    w.show()
    sys.exit(app.exec())

if __name__ == "__main__":
    main()
```

TaskAssignmentPageUI.py

```
# -*- coding: utf-8 -*-

##################################################################
↪  ##############
## Form generated from reading UI file 'TaskAssignmentPage (1).ui'
##
## Created by: Qt User Interface Compiler version 6.10.1
##
## WARNING! All changes made in this file will be lost when
↪  recompiling UI file!
##################################################################
↪  ##############

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime,
↪  QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient,
↪  QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QHBoxLayout, QLabel,
↪  QPushButton,
    QSizePolicy, QStackedWidget, QTextEdit, QVBoxLayout,
    QWidget)

class Ui_Form(object):
    def setupUi(self, Form):
        if not Form.objectName():
            Form.setObjectName(u"Form")
        Form.resize(1280, 700)
        Form.setMinimumSize(QSize(0, 0))
        self.verticalLayout = QVBoxLayout(Form)
        self.verticalLayout.setObjectName(u"verticalLayout")
        self.navbar_hbox = QHBoxLayout()
        self.navbar_hbox.setObjectName(u"navbar_hbox")
        self.logo = QLabel(Form)
        self.logo.setObjectName(u"logo")
        self.navbar_hbox.addWidget(self.logo)

        self.current_mode_label = QLabel(Form)
        self.current_mode_label.setObjectName(u"current_mode_label⌟
            ↪  ")
        font = QFont()
        font.setFamilies([u"CaskaydiaCove Nerd Font"])
        font.setPointSize(16)
        self.current_mode_label.setFont(font)

        self.navbar_hbox.addWidget(self.current_mode_label)

        self.toggle_btn = QPushButton(Form)
        self.toggle_btn.setObjectName(u"toggle_btn")
        sizePolicy = QSizePolicy(QSizePolicy.Policy.Fixed,
           ↪  QSizePolicy.Policy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.toggle_btn.sizePolicy().⌟
           ↪  hasHeightForWidth())
        self.toggle_btn.setSizePolicy(sizePolicy)
        self.toggle_btn.setMinimumSize(QSize(0, 0))
        self.toggle_btn.setFont(font)

        self.navbar_hbox.addWidget(self.toggle_btn)

        self.navbar_hbox.setStretch(0, 1)
        self.navbar_hbox.setStretch(1, 6)

        self.verticalLayout.addLayout(self.navbar_hbox)

        self.mode_tab = QStackedWidget(Form)
        self.mode_tab.setObjectName(u"mode_tab")
        self.manual_mode_tab = QWidget()
        self.manual_mode_tab.setObjectName(u"manual_mode_tab")
        self.horizontalLayout = QHBoxLayout(self.manual_mode_tab)
        self.horizontalLayout.setObjectName(u"horizontalLayout")
        self.editor_vbox = QVBoxLayout()
        self.editor_vbox.setObjectName(u"editor_vbox")
        self.code_label = QLabel(self.manual_mode_tab)
        self.code_label.setObjectName(u"code_label")
        font1 = QFont()
        font1.setPointSize(24)
        self.code_label.setFont(font1)

        self.editor_vbox.addWidget(self.code_label)

        self.editor = QTextEdit(self.manual_mode_tab)
        self.editor.setObjectName(u"editor")
        font2 = QFont()
        font2.setPointSize(16)
        font2.setKerning(False)
        self.editor.setFont(font2)
        self.editor.setStyleSheet(u"background-color: #ffffff;")

        self.editor_vbox.addWidget(self.editor)

        self.manual_btn_hbox = QHBoxLayout()
        self.manual_btn_hbox.setObjectName(u"manual_btn_hbox")
        self.manual_load_btn = QPushButton(self.manual_mode_tab)
        self.manual_load_btn.setObjectName(u"manual_load_btn")

        self.manual_btn_hbox.addWidget(self.manual_load_btn)

        self.manual_convert_btn = QPushButton(self.manual_mode_tab)
        self.manual_convert_btn.setObjectName(u"manual_convert_btn
            ↪  ")

        self.manual_btn_hbox.addWidget(self.manual_convert_btn)

        self.manual_save_btn = QPushButton(self.manual_mode_tab)
        self.manual_save_btn.setObjectName(u"manual_save_btn")

        self.manual_btn_hbox.addWidget(self.manual_save_btn)


        self.editor_vbox.addLayout(self.manual_btn_hbox)

        self.editor_vbox.setStretch(1, 1)

        self.horizontalLayout.addLayout(self.editor_vbox)
        self.preview_vbox_manual = QVBoxLayout()
        self.preview_vbox_manual.setObjectName(u"preview_vbox_manu⌋
          ↪  al")
        self.preview_label_maual = QLabel(self.manual_mode_tab)
        self.preview_label_maual.setObjectName(u"preview_label_mau⌋
          ↪  al")
        self.preview_label_maual.setFont(font1)

        self.preview_vbox_manual.addWidget(self.preview_label_maua⌋
          ↪  l)


        self.horizontalLayout.addLayout(self.preview_vbox_manual)

        self.horizontalLayout.setStretch(0, 2)
        self.horizontalLayout.setStretch(1, 3)
        self.mode_tab.addWidget(self.manual_mode_tab)
        self.sync_mode_tab = QWidget()
        self.sync_mode_tab.setObjectName(u"sync_mode_tab")
        self.horizontalLayout_4 = QHBoxLayout(self.sync_mode_tab)
        self.horizontalLayout_4.setObjectName(u"horizontalLayout_4⌋
          ↪  ")
        self.preview_vbox_sync = QVBoxLayout()
        self.preview_vbox_sync.setObjectName(u"preview_vbox_sync")
        self.preview_label_sync = QLabel(self.sync_mode_tab)
        self.preview_label_sync.setObjectName(u"preview_label_sync⌋
          ↪  ")
        self.preview_label_sync.setFont(font1)

        self.preview_vbox_sync.addWidget(self.preview_label_sync)

        self.web_engine_hbox = QHBoxLayout()
        self.web_engine_hbox.setObjectName(u"web_engine_hbox")

        self.preview_vbox_sync.addLayout(self.web_engine_hbox)

        self.sync_btn_hbox = QHBoxLayout()
        self.sync_btn_hbox.setObjectName(u"sync_btn_hbox")
        self.sync_load_btn = QPushButton(self.sync_mode_tab)
        self.sync_load_btn.setObjectName(u"sync_load_btn")
```

```
        self.sync_btn_hbox.addWidget(self.sync_load_btn)

        self.sync_convert_btn = QPushButton(self.sync_mode_tab)
        self.sync_convert_btn.setObjectName(u"sync_convert_btn")

        self.sync_btn_hbox.addWidget(self.sync_convert_btn)

        self.sync_save_btn = QPushButton(self.sync_mode_tab)
        self.sync_save_btn.setObjectName(u"sync_save_btn")

        self.sync_btn_hbox.addWidget(self.sync_save_btn)


        self.preview_vbox_sync.addLayout(self.sync_btn_hbox)

        self.preview_vbox_sync.setStretch(1, 1)

        self.horizontalLayout_4.addLayout(self.preview_vbox_sync)

        self.mode_tab.addWidget(self.sync_mode_tab)

        self.verticalLayout.addWidget(self.mode_tab)


        self.retranslateUi(Form)

        self.mode_tab.setCurrentIndex(0)


        QMetaObject.connectSlotsByName(Form)
    # setupUi

    def retranslateUi(self, Form):
        Form.setWindowTitle(QCoreApplication.translate("Form",
          ↪  u"Form", None))
        self.logo.setText("")
        self.current_mode_label.setText(QCoreApplication.translate⌋
          ↪  ("Form", u"Current Mode:", None))
        self.toggle_btn.setText(QCoreApplication.translate("Form",
          ↪  u"Toggle Mode", None))
        self.code_label.setText(QCoreApplication.translate("Form",
          ↪  u"Code:", None))
        self.editor.setPlaceholderText("")
        self.manual_load_btn.setText(QCoreApplication.translate("F⌋
          ↪  orm", u"Load", None))
        self.manual_convert_btn.setText(QCoreApplication.translate⌋
          ↪  ("Form", u"Convert", None))
        self.manual_save_btn.setText(QCoreApplication.translate("F⌋
          ↪  orm", u"Save", None))
        self.preview_label_maual.setText(QCoreApplication.translat⌋
          ↪  e("Form", u"Preview:", None))
        self.preview_label_sync.setText(QCoreApplication.translate⌋
          ↪  ("Form", u"Preview:", None))
        self.sync_load_btn.setText(QCoreApplication.translate("For⌋
          ↪  m", u"Load", None))
        self.sync_convert_btn.setText(QCoreApplication.translate("⌋
          ↪  Form", u"Convert", None))
        self.sync_save_btn.setText(QCoreApplication.translate("For⌋
          ↪  m", u"Save", None))
    # retranslateUi
```

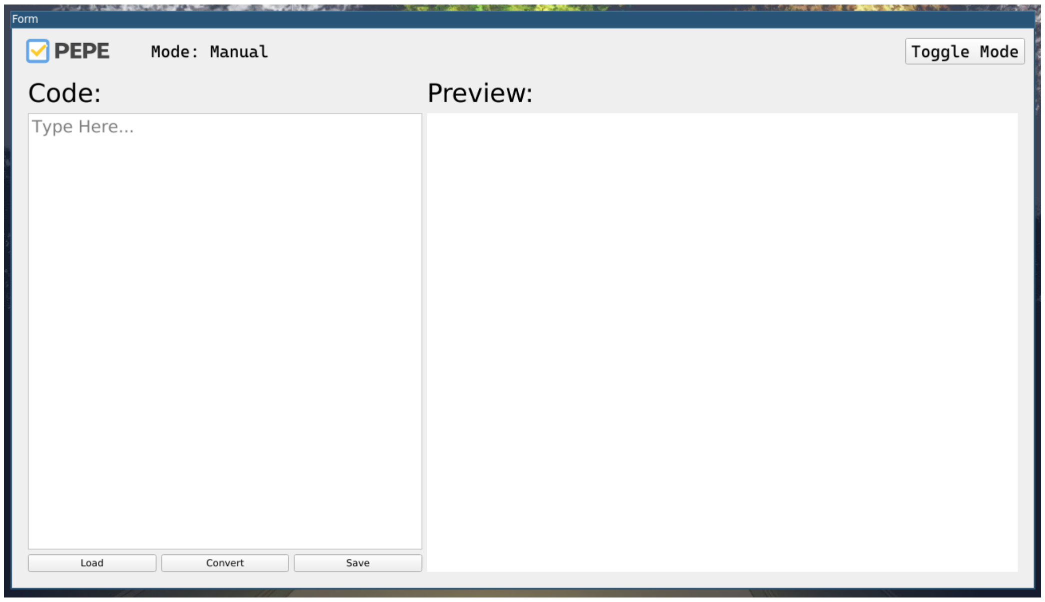
Image:

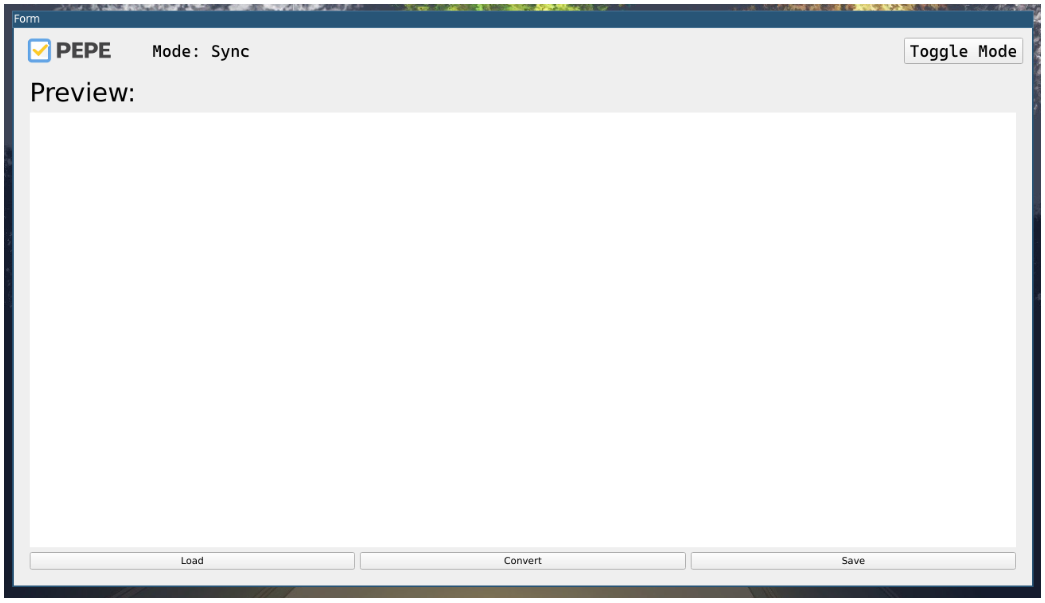
logo.png

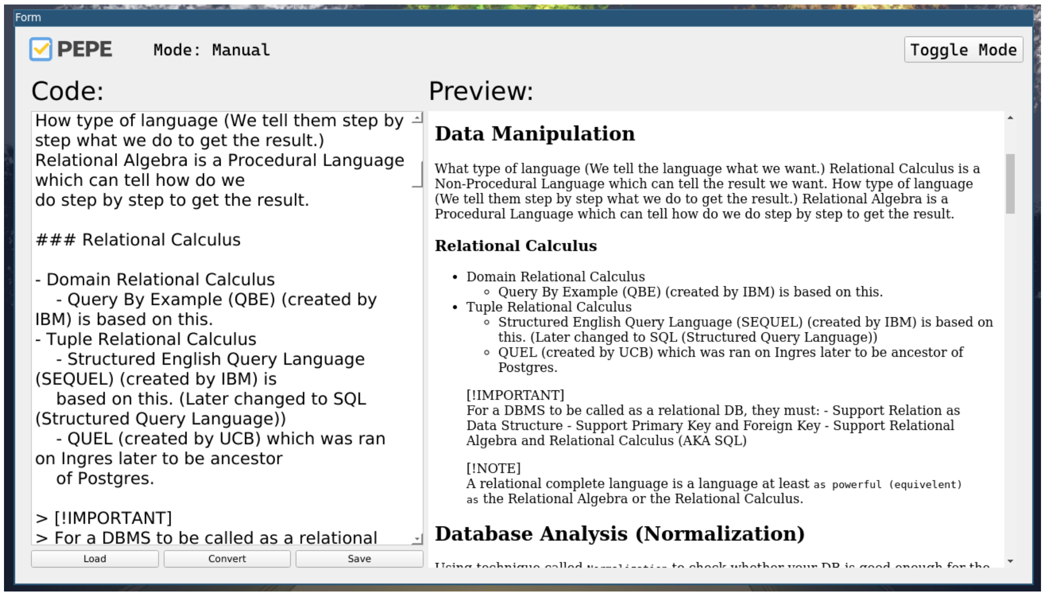
PEPE

Output:

Task Assignment Page







```
How type of language (We tell them step by
step what we do to get the result.)
Relational Algebra is a Procedural Language
which can tell how do we
do step by step to get the result.

### Relational Calculus

- Domain Relational Calculus
    - Query By Example (QBE) (created by
IBM) is based on this.
- Tuple Relational Calculus
    - Structured English Query Language
(SEQUEL) (created by IBM) is
    based on this. (Later changed to SQL
(Structured Query Language))
    - QUEL (created by UCB) which was ran
on Ingres later to be ancestor
    of Postgres.

> [!IMPORTANT]
> For a DBMS to be called as a relational
```

Data Manipulation

What type of language (We tell the language what we want.) Relational Calculus is a Non-Procedural Language which can tell the result we want. How type of language (We tell them step by step what we do to get the result.) Relational Algebra is a Procedural Language which can tell how do we do step by step to get the result.

Relational Calculus

- Domain Relational Calculus
  - Query By Example (QBE) (created by IBM) is based on this.
- Tuple Relational Calculus
  - Structured English Query Language (SEQUEL) (created by IBM) is based on this. (Later changed to SQL (Structured Query Language))
  - QUEL (created by UCB) which was ran on Ingres later to be ancestor of Postgres.

[!IMPORTANT]
For a DBMS to be called as a relational DB, they must: - Support Relation as Data Structure - Support Primary Key and Foreign Key - Support Relational Algebra and Relational Calculus (AKA SQL)

[!NOTE]
A relational complete language is a language at least `as powerful (equivelent) as` the Relational Algebra or the Relational Calculus.

Database Analysis (Normalization)

Using technique called `Normalization` to check whether your DB is good enough for the

External Plugin

- `pip install markdown`