Homework 9

Software Engineering Principle

Software Engineering Program,

Department of Computer Engineering,

School of Engineering, KMITL

67011287 Ramida Laphasphokin

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

- **Design Page**: Page to brainstorm the program architecture and to adjust tasks duration or deadline.

Code:

HW9_67011287_Ramida.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
      ↪  initial-scale=1">

    <title>Basic PyScript!</title>

    <link rel="stylesheet"
          href="https://pyscript.net/releases/2026.1.1/core.css" />
    <script type="module"
            src="https://pyscript.net/releases/2026.1.1/core.js"></
              ↪  /script>
</head>

<body style="margin: 0; padding: 0;">
    <section class="pyscript">
        <div id="container"></div>

        <script type="py" src="./HW9_67011287_Ramida.py"></script>
    </section>
</body>
</html>
```

HW9_67011287_Ramida.py

```
import js
from pyscript import document
from pyodide.ffi import create_proxy
from abc import ABC, abstractmethod


class AbstractWidget(ABC):
    def __init__(self, element_id):
        self.element_id = element_id
        self._element = None

    @property
    def element(self):
        """Return the DOM container element"""
        if not self._element:
            self._element =
              ↪  document.querySelector(f"#{self.element_id}")
        return self._element

    @abstractmethod
    def drawWidget(self):
        pass


class Design(AbstractWidget):
    def __init__(self, element_id):
        super().__init__(element_id)

        self.pages = {}
        self.current_page = None

        self.hover_sound = "./sounds/hover.wav"
        self.click_sound = "./sounds/page_toggle.wav"

        self.diagram_images = {
            "class": "./images/class_diagram.png",
            "interaction": "./images/interaction_diagram.png",
            "gantt": "./images/gantt_chart.png"
        }
```

```
def switch_page(self, page_name):
    for _, page in self.pages.items():
        page.style.display = "none"

    self.pages[page_name].style.display = "flex"
    self.current_page = page_name

def on_nav_click(self, page_name):
    def handler(event):
        js.Audio.new(self.click_sound).play()
        self.switch_page(page_name)

    return handler

def on_nav_hover(self):
    js.Audio.new(self.hover_sound).play()

def create_nav_button(self, text, page_name):
    btn = document.createElement("button")
    btn.innerText = text

    btn.style.flex = "1"
    btn.style.border = "none"
    btn.style.background = "#2f8ed2"
    btn.style.color = "white"
    btn.style.fontSize = "16px"
    btn.style.cursor = "pointer"
    btn.style.padding = "20px"

    btn.onmouseenter = create_proxy(self.on_nav_hover)
    btn.onmouseenter = lambda e: (
        self.on_nav_hover(),
        btn.style.setProperty("background", "#fdd148"),
        btn.style.setProperty("color", "black")
    )

    btn.onmouseleave = lambda e: (
        btn.style.setProperty("background", "#2f8ed2"),
        btn.style.setProperty("color", "white")
    )
```

```
        btn.onclick = create_proxy(self.on_nav_click(page_name))

        return btn

    def create_sidebar(self):
        sidebar = document.createElement("div")

        sidebar.style.width = "170px"
        sidebar.style.background = "#d9ecf7"
        sidebar.style.borderRight = "1px solid #b0cddd"
        sidebar.style.display = "flex"
        sidebar.style.flexDirection = "column"
        sidebar.style.padding = "15px"
        sidebar.style.boxSizing = "border-box"

        title = document.createElement("h3")
        title.innerText = "Tools:"
        sidebar.appendChild(title)

        for tool in ["Tool 1", "Tool 2", "Tool 3"]:
            btn = document.createElement("button")
            btn.innerText = tool

            btn.style.marginTop = "10px"
            btn.style.padding = "10px"
            btn.style.border = "none"
            btn.style.borderRadius = "8px"
            btn.style.cursor = "pointer"
            btn.style.fontSize = "14px"
            btn.style.background = "white"

            btn.onmouseenter = lambda e, b=btn:
               ↪  b.style.setProperty(
                "background", "#fdd148"
            )
            btn.onmouseleave = lambda e, b=btn:
               ↪  b.style.setProperty(
                "background", "white"
            )
```

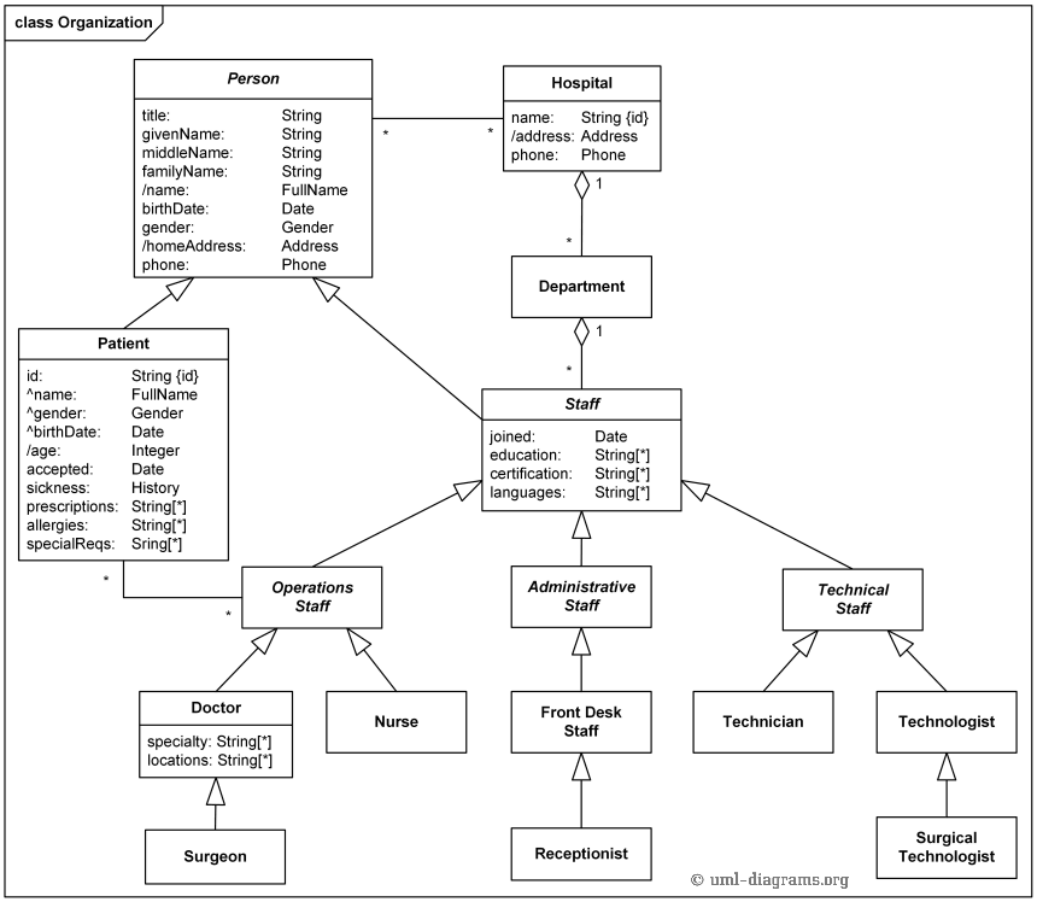
```
            sidebar.appendChild(btn)

        return sidebar

    def create_diagram_area(self, page_name, title_text):
        container = document.createElement("div")
        container.style.flex = "1"
        container.style.display = "flex"
        container.style.flexDirection = "column"
        container.style.padding = "20px"
        container.style.boxSizing = "border-box"

        title = document.createElement("h2")
        title.innerText = title_text
        title.style.textAlign = "center"
        title.style.marginBottom = "15px"

        img = document.createElement("img")
        img.src = self.diagram_images[page_name]

        img.style.flex = "1"
        img.style.width = "100%"
        img.style.height = "100%"
        img.style.objectFit = "contain"
        img.style.border = "2px dashed #999"
        img.style.background = "#eeeeee"
        img.style.borderRadius = "12px"

        container.appendChild(title)
        container.appendChild(img)

        return container

    def create_page(self, page_name, title_text, sidebar=True):
        page = document.createElement("div")
        page.style.flex = "1"
        page.style.display = "flex"
        page.style.width = "100%"
        page.style.height = "100%"

        if sidebar:
```

```
            page.appendChild(self.create_sidebar())

        page.appendChild(self.create_diagram_area(page_name,
            ↪ title_text))

        self.pages[page_name] = page
        return page

    def drawWidget(self):
        root = document.createElement("div")
        root.style.width = "100%"
        root.style.height = "100vh"
        root.style.display = "flex"
        root.style.flexDirection = "column"
        root.style.background = "#f7f7f7"

        navbar = document.createElement("div")
        navbar.style.display = "flex"
        navbar.style.height = "90px"
        navbar.style.background = "#2f8ed2"

        navbar.appendChild(self.create_nav_button("Class",
            ↪ "class"))
        navbar.appendChild(self.create_nav_button("Interaction",
            ↪ "interaction"))
        navbar.appendChild(self.create_nav_button("Gantt Chart",
            ↪ "gantt"))

        root.appendChild(navbar)

        content = document.createElement("div")
        content.style.flex = "1"
        content.style.display = "flex"

        content.appendChild(self.create_page("class", "Class
            ↪ Diagram", sidebar=True))
        content.appendChild(self.create_page("interaction",
            ↪ "Interaction Diagram", sidebar=True))
        content.appendChild(self.create_page("gantt", "Gantt
            ↪ Chart", sidebar=False))
```

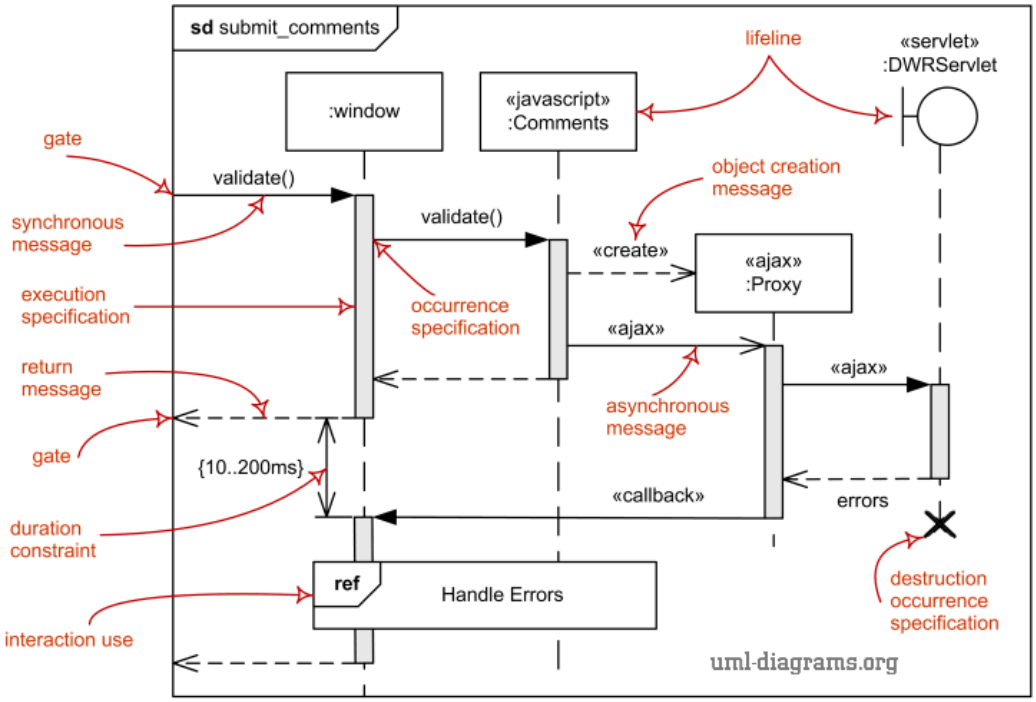
```
        root.appendChild(content)

        self.element.appendChild(root)

        self.switch_page("class")

if __name__ == "__main__":
    ui = Design("container")
    ui.drawWidget()
```

Image:

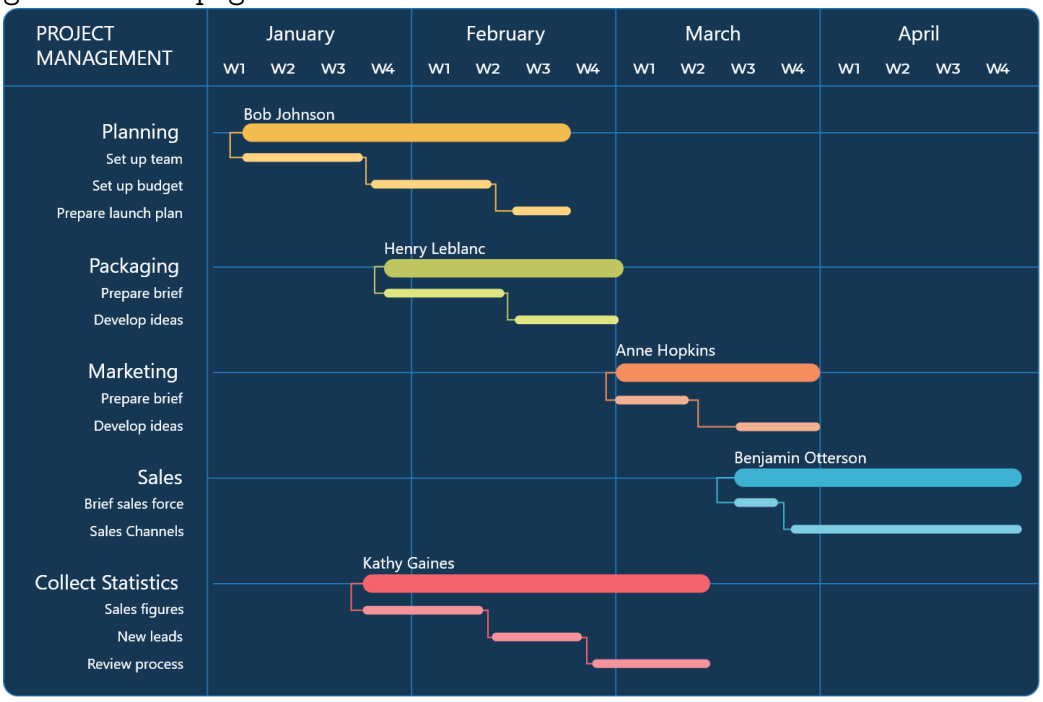
`class_diagram.png`



interaction_diagram.png



gantt_chart.png

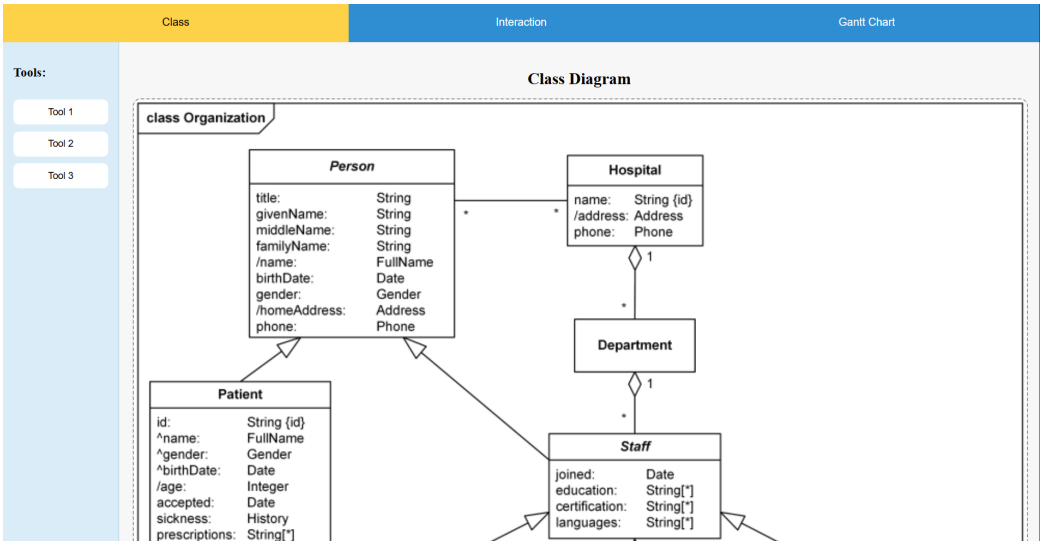


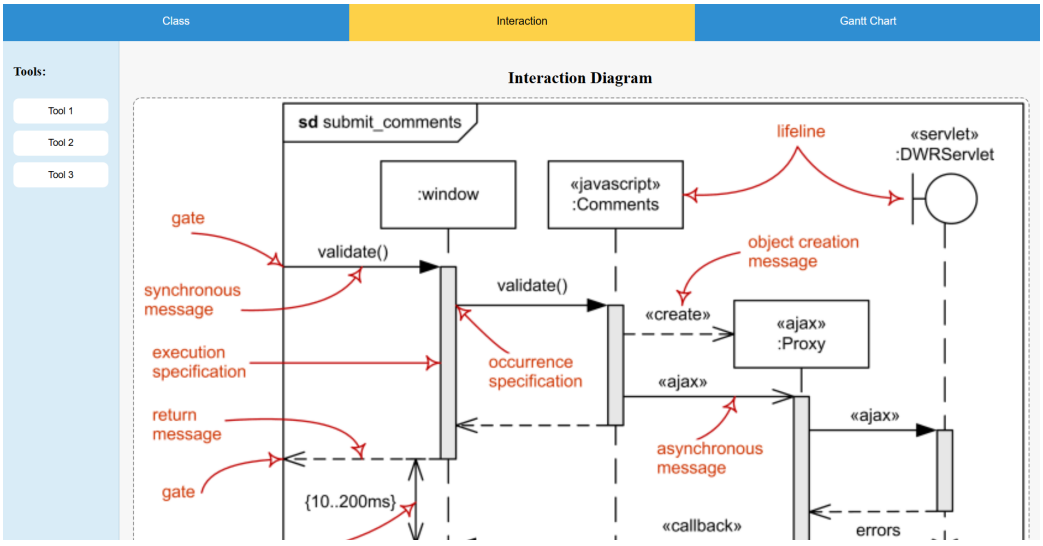
Sound:

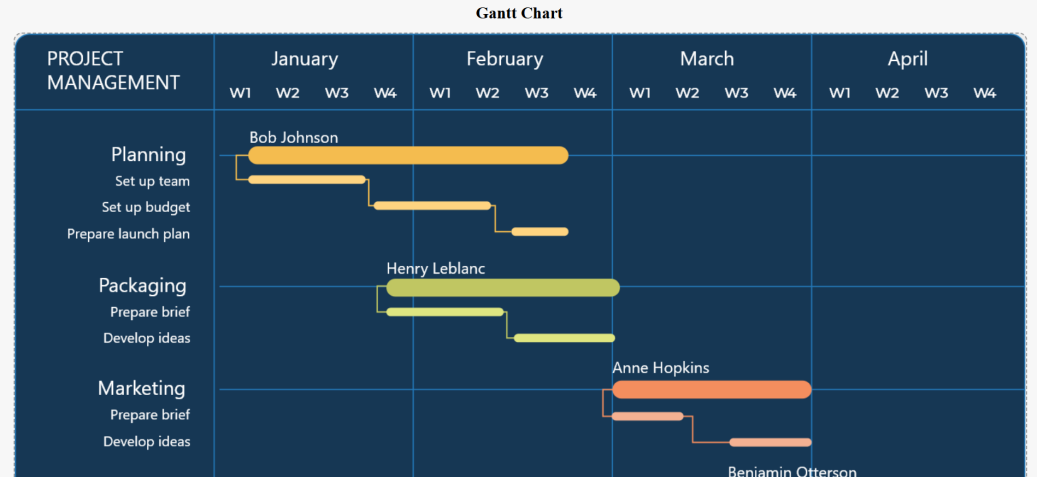
- hover.wav
- page_toggle.wav

Output:

Design Page






Gantt Chart
PROJECT MANAGEMENT
January
February
March
April
W1 W2 W3 W4 W1 W2 W3 W4 W1 W2 W3 W4 W1 W2 W3 W4
Planning
Set up team
Set up budget
Prepare launch plan
Bob Johnson
Packaging
Prepare brief
Develop ideas
Henry Leblanc
Marketing
Prepare brief
Develop ideas
Anne Hopkins
Benjamin Otterson