Kế hoạch học tập: Xây dựng Quiz App bằng Kivy

Tài liệu này dành cho học sinh THCS, giúp các em từng bước tự xây dựng ứng dụng Quiz App (trắc nghiệm) bằng Python + Kivy theo mô hình MVC.

---

1. Tổng quan (Overview)

Mục tiêu cuối cùng: Học sinh tạo được một ứng dụng Quiz App đơn giản với 3 màn hình:

- Màn hình bắt đầu (Start)
- Màn hình làm bài (Quiz)
- Màn hình kết quả (Result)

Ứng dụng chia theo mô hình **MVC**:

- **Model**: dữ liệu câu hỏi + tính điểm
- **View**: giao diện Kivy (các Screen)
- **Controller**: điều khiển luồng làm bài, nối Model và View

Kiến thức nền tảng cần có trước:

- Biết chạy file Python (`python main.py`)
- Biết kiểu dữ liệu cơ bản: `int` , `str` , `list` , `dict`
- Biết dùng `if` , `for` , hàm (`def ...`)

---

2. Sơ đồ thư mục mục tiêu

Học sinh sẽ dần dần tạo được cấu trúc thư mục như sau:

```text
main.py
models/
    quiz_data.py
    quiz_model.py
views/
    widgets.py
    start_screen.py
    quiz_screen.py
    result_screen.py
controllers/
    quiz_controller.py
```

---

3. Các task nhỏ cho học sinh

Task 1: Chuẩn bị môi trường Kivy

Mục tiêu: Chạy được một cửa sổ Kivy đơn giản.

- Viết file `hello_kivy.py` hiển thị một Label đơn giản.
- Chạy thử và kiểm tra có cửa sổ hiện ra.

Keyword / Kiến thức cần:

- `from kivy.app import App`
- `from kivy.uix.label import Label`
- Lớp `App`, hàm `build`, `App().run()`

💡 **Gợi ý**:

- Nếu chưa cài Kivy, chạy: `pip install kivy`
- Cửa sổ Kivy mặc định có kích thước 800x600 pixels
- Nếu gặp lỗi, thử chạy lại hoặc kiểm tra phiên bản Python (nên dùng 3.8+)

📝 **Các bước thực hiện**:

1. Tạo file mới tên `hello_kivy.py`
2. Import thư viện cần thiết:

```python
from kivy.app import App
from kivy.uix.label import Label
```

3. Tạo class kế thừa từ `App`:

```python
class HelloApp(App):
    def build(self):
        return Label(text="Hello Kivy!")
```

4. Thêm dòng chạy ứng dụng:

```python
if __name__ == "__main__":
    HelloApp().run()
```

5. Chạy thử: `python hello_kivy.py`

6. Kiểm tra: Cửa sổ có hiện chữ "Hello Kivy!" không?

Task 2: Hiểu mô hình MVC ở cấp độ đơn giản

Mục tiêu: Học sinh hiểu 3 phần chính của ứng dụng:

- Model = dữ liệu + tính toán điểm
- View = giao diện (màn hình, nút bấm, label)
- Controller = điều khiển (khi bấm nút thì làm gì)

Việc cần làm:

- Vẽ sơ đồ đơn giản (trên giấy hoặc trong vở):
  - View → App → Controller → Model → Controller → View
- Ghi chú 1–2 ví dụ: "khi bấm đáp án A thì chuyện gì xảy ra?"

Keyword / Kiến thức cần:

- Khái niệm **Model – View – Controller**
- Hàm được gọi khi bấm nút: `on_press`

💡 **Gợi ý**:

- **Model** giống như "bộ não" - lưu trữ và xử lý dữ liệu
- **View** giống như "khuôn mặt" - cái mà người dùng nhìn thấy
- **Controller** giống như "bộ phận điều khiển" - kết nối giữa não và mặt

📝 **Các bước thực hiện**:

1. Vẽ sơ đồ với 3 hộp: Model, View, Controller

2. Vẽ mũi tên kết nối:
    - View → Controller (người dùng bấm nút)
    - Controller → Model (hỏi câu trả lời đúng không?)
    - Model → Controller (trả lời: đúng hoặc sai)
    - Controller → View (cập nhật màu nút, điểm số)
3. Viết ví dụ cụ thể:
    - "Khi bấm đáp án A → Controller kiểm tra → Model tính điểm → View hiện kết quả"
4. Thảo luận: Tại sao phải tách ra 3 phần thay vì viết tất cả trong 1 file?

Task 3: Tạo dữ liệu câu hỏi (Model – phần 1)

Mục tiêu: Tạo được danh sách câu hỏi dạng `list` chứa `dict`.

Việc cần làm:

- Tạo thư mục `models/`.
- Tạo file `models/quiz_data.py`.
- Tạo biến `QUIZ_QUESTIONS` là **list** các câu hỏi, mỗi câu hỏi là **dict** có dạng:

```python
{
    "question": "Câu hỏi gì đó?",
    "choices": ["A", "B", "C", "D"],
    "answer_index": 0,  # vị trí đáp án đúng
}
```

Keyword / Kiến thức cần:

- `list` và `dict` trong Python
- Truy cập `dict["key"]`

💡 **Gợi ý**:

- Bắt đầu với 5-10 câu hỏi đơn giản về môn học yêu thích
- `answer_index` bắt đầu từ 0 (0=A, 1=B, 2=C, 3=D)
- Có thể copy cấu trúc dict đầu tiên và sửa nội dung cho nhanh

📝 **Các bước thực hiện**:

1. Tạo thư mục `models` trong thư mục dự án
2. Tạo file `models/quiz_data.py`
3. Thêm comment mô tả:

```python
# models/quiz_data.py
# Chứa danh sách câu hỏi cho quiz
```

4. Tạo list bắt đầu:

```python
QUIZ_QUESTIONS = [
```

5. Thêm câu hỏi đầu tiên (ví dụ):

```python
{
    "question": "Thủ đô của Việt Nam là?",
    "choices": ["Hà Nội", "Huế", "Đà Nẵng", "TP.HCM"],
    "answer_index": 0,
},
```

6. Thêm 4-9 câu hỏi khác (có thể về Toán, Tin, Địa lý...)

7. Đóng list: `]`

8. Test: Mở Python console và chạy:

```python
from models.quiz_data import QUIZ_QUESTIONS
print(QUIZ_QUESTIONS[0])  # In câu hỏi đầu tiên
```

Task 4: Xây dựng QuizModel (Model – phần 2)

Mục tiêu: Biết lưu trạng thái bài quiz (câu hiện tại, điểm) và kiểm tra đáp án.

Việc cần làm:

- Tạo file `models/quiz_model.py`.
- Viết lớp `QuizModel` với các thuộc tính:
  - `self.all_questions` – danh sách câu hỏi lấy từ `QUIZ_QUESTIONS`
  - `self.total_questions` – số câu cho học sinh làm (ví dụ 5)
  - `self.current_index` – chỉ số câu hiện tại
  - `self.score` – số câu đúng

- Thêm các hàm đơn giản:
  - `reset()` – đặt lại `current_index` về 0, `score` về 0
  - `get_active_questions()` – trả về N câu hỏi đầu
  - `get_current_question()` – trả về câu hỏi hiện tại
  - `check_answer(choice_index)` – kiểm tra đúng/sai, tăng điểm nếu đúng
  - `next_question()` – tăng `current_index`

Keyword / Kiến thức cần:

- Lớp (`class`), hàm khởi tạo `__init__`
- Thuộc tính đối tượng `self.xxx`

💡 **Gợi ý:**

- `current_index` bắt đầu từ 0, tương ứng với câu hỏi đầu tiên trong list
- Hàm `check_answer` cần so sánh `choice_index` với `answer_index` của câu hỏi
- Dùng `self.all_questions[:]` để copy list, tránh thay đổi dữ liệu gốc

📝 **Các bước thực hiện:**

1. Tạo file `models/quiz_model.py`
2. Import dữ liệu:

```python
from .quiz_data import QUIZ_QUESTIONS
```

3. Tạo class và hàm `__init__` :

```python
class QuizModel:
    def __init__(self, total_questions=5):
        self.all_questions = QUIZ_QUESTIONS[:]
        self.total_questions = min(total_questions, len(self.all_questior
        self.current_index = 0
        self.score = 0
```

4. Viết hàm `reset()` :

```python
def reset(self):
    self.current_index = 0
    self.score = 0
```

5. Viết hàm `get_active_questions()` - lấy N câu đầu:

```python
def get_active_questions(self):
    return self.all_questions[:self.total_questions]
```

6. Viết hàm `get_current_question()` - lấy câu hiện tại:

```python
def get_current_question(self):
    if self.current_index >= len(self.get_active_questions()):
        return None
    return self.get_active_questions()[self.current_index]
```

7. Viết hàm `check_answer(choice_index)` - kiểm tra và tính điểm:

```python
def check_answer(self, choice_index):
    question = self.get_current_question()
    if question is None:
        return False
    correct = (choice_index == question["answer_index"])
    if correct:
        self.score += 1
    return correct
```

8. Viết hàm `next_question()` - chuyển câu tiếp:

```python
def next_question(self):
    self.current_index += 1
```

9. Test đơn giản:

```python
if __name__ == "__main__":
    model = QuizModel()
    print(model.get_current_question())
    print(model.check_answer(0))  # Test đáp án đầu tiên
```

Task 5: Tạo các widget cơ bản (View – phần 1)

Mục tiêu: Tạo các hàm helper để làm giao diện gọn và thống nhất.

Việc cần làm:

- Tạo thư mục `views/` .
- Tạo file `views/widgets.py` .
- Viết các hàm:
  - `make_title(text)` – trả về Label tiêu đề
  - `make_subtitle(text)` – trả về Label mô tả/hướng dẫn
  - `make_primary_button(text, on_press)` – trả về Button chính
  - `make_answer_button(prefix)` – trả về Button đáp án
- Sau đó dùng thử trong một file nhỏ để thấy kết quả.

Keyword / Kiến thức cần:

- `Label` , `Button` trong Kivy
- Tham số `on_press` và `btn.bind(on_press=...)`

💡 **Gợi ý**:

- Dùng `font_size` để điều chỉnh kích thước chữ (vd: "24sp", "16sp")
- Màu sắc trong Kivy dùng định dạng RGBA với giá trị 0-1: `(0.2, 0.6, 1, 1)`
- `size_hint` giúp widget tự điều chỉnh kích thước theo màn hình

📝 **Các bước thực hiện**:

1. Tạo thư mục `views`
2. Tạo file `views/widgets.py`
3. Import thư viện:

```python
from kivy.uix.button import Button
from kivy.uix.label import Label
from kivy.metrics import dp
```

4. Định nghĩa các màu (có thể thay đổi tùy ý):

```python
PRIMARY_COLOR = (0.2, 0.6, 1, 1)  # Màu xanh dương
BACKGROUND_COLOR = (0.95, 0.95, 0.95, 1)  # Màu xám nhạt
```

5. Viết hàm `make_title(text)` :

```python
def make_title(text):
    return Label(
        text=text,
        font_size="28sp",
        bold=True,
        size_hint_y=None,
        height=dp(50)
    )
```

6. Viết hàm `make_subtitle(text)` (tương tự nhưng font nhỏ hơn)
7. Viết hàm `make_primary_button(text, on_press)`:

```python
def make_primary_button(text, on_press):
    btn = Button(
        text=text,
        size_hint=(0.6, None),
        height=dp(50),
        background_color=PRIMARY_COLOR
    )
    btn.bind(on_press=on_press)
    return btn
```

8. Viết hàm `make_answer_button(prefix)` (Button cho đáp án A/B/C/D)
9. Test trong file riêng hoặc thử import vào `hello_kivy.py` để xem kết quả

Task 6: Xây dựng StartScreen (View – phần 2)

Mục tiêu: Tạo màn hình bắt đầu với tên app, hướng dẫn và nút "BẮT ĐẦU".

Việc cần làm:

- Tạo file `views/start_screen.py`.
- Tạo lớp `StartScreen(Screen)`.
- Dùng `BoxLayout` dọc, thêm:
  - Tiêu đề app (dùng `make_title`)
  - Label mô tả ngắn (phiên bản, ghi chú)
  - Hướng dẫn (dùng `make_subtitle`)
  - Nút "BẮT ĐẦU" → gọi `app.start_quiz()` khi bấm.

Keyword / Kiến thức cần:

- `Screen` , `BoxLayout`
- `App.get_running_app()`

💡 **Gợi ý**:

- `BoxLayout(orientation="vertical")` sắp xếp các widget từ trên xuống dưới
- Dùng `Label()` trống để tạo khoảng trắng (spacer)
- `App.get_running_app()` giúp gọi hàm từ class chính

📝 **Các bước thực hiện**:

1. Tạo file `views/start_screen.py`
2. Import các thư viện cần thiết:

```python
from kivy.app import App
from kivy.uix.screenmanager import Screen
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from .widgets import make_title, make_subtitle, make_primary_button
```

3. Tạo class `StartScreen(Screen)` :

```python
class StartScreen(Screen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
```

4. Tạo layout chính:

```python
layout = BoxLayout(orientation="vertical", padding=20, spacing=15)
```

5. Thêm các widget theo thứ tự:
    - `layout.add_widget(Label())` # spacer trên
    - `layout.add_widget(make_title("QUIZ APP"))`
    - `layout.add_widget(make_subtitle("Phiên bản 1.0"))`
    - `layout.add_widget(make_subtitle("Chọn đáp án đúng cho mỗi câu hỏi"))`
    - `layout.add_widget(Label())` # spacer giữa
6. Định nghĩa hàm xử lý nút:

```python
def on_start_pressed(btn):
    app = App.get_running_app()
    app.start_quiz()
```

7. Thêm nút bắt đầu:

```python
layout.add_widget(make_primary_button("BẮT ĐẦU", on_start_pressed))
```

8. Thêm spacer dưới và gắn layout:

```python
layout.add_widget(Label())
self.add_widget(layout)
```

Task 7: Xây dựng QuizScreen (View – phần 3)

Mục tiêu: Tạo màn hình câu hỏi với 4 đáp án, hiện số câu và điểm hiện tại.

Việc cần làm:

- Tạo file `views/quiz_screen.py` .
- Tạo lớp `QuizScreen(Screen)` .
- Bên trong:
  - Label trên cùng: "Câu X/Y Điểm: Z"
  - Label hiển thị nội dung câu hỏi
  - 4 Button đáp án A/B/C/D (dùng `make_answer_button`)
  - Label nhỏ đưa feedback (Đúng / Sai)
  - Nút "CÂU TIẾP"
- Khi học sinh chọn đáp án:
  - Gọi `app.check_answer(choice_index)`
  - Hiện feedback đúng/sai
  - Cho phép bấm "CÂU TIẾP" để sang câu tiếp theo.

Keyword / Kiến thức cần:

- `GridLayout` (hiển thị danh sách đáp án theo cột)
- Event `on_press` cho từng nút đáp án

💡 **Gợi ý**:

- Dùng `self.selected = False` để theo dõi xem đã chọn đáp án chưa
- `GridLayout(cols=1)` tạo lưới 1 cột, các button xếp dọc
- Vô hiệu hóa các nút sau khi chọn bằng `btn.disabled = True`
- Dùng màu xanh cho đúng, màu đỏ cho sai

📝 **Các bước thực hiện:**

1. Tạo file `views/quiz_screen.py`

2. Import:

```python
from kivy.app import App
from kivy.uix.screenmanager import Screen
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.gridlayout import GridLayout
from kivy.uix.label import Label
from .widgets import make_primary_button, make_answer_button
```

3. Tạo class với biến theo dõi:

```python
class QuizScreen(Screen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.selected = False
```

4. Tạo layout chính và các label:

```python
layout = BoxLayout(orientation="vertical", padding=20, spacing=10)
self.top_info = Label(text="Câu 1/5    Điểm: 0", size_hint_y=None, height
self.question_label = Label(text="Câu hỏi sẽ hiện ở đây", font_size="18sp
```

5. Tạo lưới đáp án với 4 nút:

```python
self.answers_grid = GridLayout(cols=1, spacing=10, size_hint_y=None)
self.answer_buttons = []
for i in range(4):
    btn = make_answer_button(["A", "B", "C", "D"][i])
    btn.bind(on_press=self.make_answer_handler(i))
    self.answer_buttons.append(btn)
    self.answers_grid.add_widget(btn)
```

6. Viết hàm xử lý khi chọn đáp án:

```python
def make_answer_handler(self, choice_index):
    def handler(btn):
        if self.selected:
            return
        self.selected = True
        app = App.get_running_app()
        is_correct = app.check_answer(choice_index)
        if is_correct:
            self.feedback_label.text = " Đúng rồi!"
            btn.background_color = (0, 1, 0, 1)  # Xanh lá
        else:
            self.feedback_label.text = " Sai rồi!"
            btn.background_color = (1, 0, 0, 1)  # Đỏ
        self.next_btn.disabled = False
    return handler
```

7. Thêm label feedback và nút "CÂU TIẾP"
8. Viết hàm `show_question(question_data, q_index, total, score)` để cập nhật giao diện

Task 8: Xây dựng ResultScreen (View – phần 4)

Mục tiêu: Hiện điểm và xếp loại sau khi làm xong.

Việc cần làm:

- Tạo file `views/result_screen.py` .
- Tạo lớp `ResultScreen(Screen)` .
- Thêm:
  - Tiêu đề "KẾT QUẢ"
  - Label: "Bạn đúng X/Y"

- Label: "Xếp loại: ..." (dựa trên điểm)
- Nút "CHƠI LẠI" → gọi `app.restart_quiz()`
- Nút "THOÁT" → đóng app

Keyword / Kiến thức cần:

- Viết hàm `show_result(score, total)` để cập nhật kết quả trên màn hình
- Decision đơn giản với `if/elif/else` để xếp loại

💡 **Gợi ý**:

- Xếp loại theo phần trăm: >= 80% = Xuất sắc, >= 60% = Khá, >= 40% = Trung bình
- Dùng `App.get_running_app().stop()` để thoát ứng dụng
- Có thể thêm emoji để màn hình sinh động hơn (🎉, 😊, 😢)

📝 **Các bước thực hiện**:

1. Tạo file `views/result_screen.py`
2. Import thư viện:

```python
from kivy.app import App
from kivy.uix.screenmanager import Screen
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from .widgets import make_title, make_primary_button
```

3. Tạo class với layout:

```python
class ResultScreen(Screen):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        layout = BoxLayout(orientation="vertical", padding=20, spacing=15
```

4. Thêm tiêu đề:

```python
layout.add_widget(make_title("KẾT QUẢ"))
```

5. Tạo các label để hiển thị kết quả:

```python
self.score_label = Label(text="", font_size="20sp")
self.rank_label = Label(text="", font_size="18sp")
layout.add_widget(self.score_label)
layout.add_widget(self.rank_label)
```

6. Viết hàm `show_result(score, total)` :

```python
def show_result(self, score, total):
    self.score_label.text = f"Bạn đúng {score}/{total} câu"
    percent = (score / total) * 100
    if percent >= 80:
        rank = "Xuất sắc "
    elif percent >= 60:
        rank = "Khá "
    elif percent >= 40:
        rank = "Trung bình "
    else:
        rank = "Cần cố gắng thêm "
    self.rank_label.text = f"Xếp loại: {rank}"
```

7. Thêm nút "CHƠI LẠI":

```python
def on_replay(btn):
    app = App.get_running_app()
    app.restart_quiz()
layout.add_widget(make_primary_button("CHƠI LẠI", on_replay))
```

8. Thêm nút "THOÁT":

```python
def on_exit(btn):
    App.get_running_app().stop()
layout.add_widget(make_primary_button("THOÁT", on_exit))
```

Task 9: Viết QuizController (Controller)

Mục tiêu: Nối Model và View, điều khiển luồng làm bài.

Việc cần làm:

- Tạo thư mục `controllers/` (nếu chưa có).
- Tạo file `controllers/quiz_controller.py` .
- Tạo lớp `QuizController` với các thuộc tính:
  - `self.model`
  - `self.sm` (ScreenManager)
  - `self.quiz_screen` , `self.result_screen`
- Thêm các hàm:
  - `start_quiz()` – gọi `model.reset()` rồi hiển thị câu đầu tiên.
  - `restart_quiz()` – gọi `start_quiz()` và chuyển sang màn "quiz".
  - `check_answer(choice_index)` – gọi `model.check_answer(...)` .
  - `next_question()` – gọi `model.next_question()` và hiển thị lại.
  - `_show_current_question()` – nếu hết câu thì chuyển sang màn kết quả; nếu chưa thì gọi `quiz_screen.show_question(...)` .

Keyword / Kiến thức cần:

- Truyền đối tượng (model, screen manager, screens) vào `__init__`
- Gọi hàm giữa các đối tượng khác nhau

💡 **Gợi ý**:

- Controller là cầu nối, không chứa logic phức tạp
- Mỗi hàm trong Controller thường chỉ gọi 1-2 hàm từ Model hoặc View
- Dùng `self.sm.current = "quiz"` để chuyển màn hình

📝 **Các bước thực hiện**:

1. Tạo thư mục `controllers`
2. Tạo file `controllers/quiz_controller.py`
3. Tạo class với `__init__` :

```python
class QuizController:
    def __init__(self, model, screen_manager, quiz_screen, result_screen)
        self.model = model
        self.sm = screen_manager
        self.quiz_screen = quiz_screen
        self.result_screen = result_screen
```

4. Viết hàm `start_quiz()` :

```python
def start_quiz(self):
    self.model.reset()
    self.sm.current = "quiz"
    self._show_current_question()
```

5. Viết hàm `check_answer(choice_index)`:

```python
def check_answer(self, choice_index):
    return self.model.check_answer(choice_index)
```

6. Viết hàm `next_question()`:

```python
def next_question(self):
    self.model.next_question()
    self._show_current_question()
```

7. Viết hàm `_show_current_question()` - phần quan trọng:

```python
def _show_current_question(self):
    question = self.model.get_current_question()
    if question is None:  # Hết câu hỏi
        score, total = self.model.get_summary()
        self.result_screen.show_result(score, total)
        self.sm.current = "result"
    else:  # Còn câu
        q_index = self.model.current_index + 1
        total = len(self.model.get_active_questions())
        score = self.model.score
        self.quiz_screen.show_question(question, q_index, total, score)
```

8. Viết hàm `restart_quiz()`:

```python
def restart_quiz(self):
    self.start_quiz()
```

Task 10: Ghép tất cả trong main.py

Mục tiêu: Tạo điểm khởi chạy chính của app.

Việc cần làm:

- Trong `main.py` :
  - Tạo `ScreenManager` với 3 màn hình: Start, Quiz, Result
  - Tạo `QuizModel`
  - Tạo `QuizController` và truyền các màn hình vào
  - Trong lớp `QuizApp` , viết các hàm:
    - `start_quiz()` → gọi `controller.start_quiz()`
    - `restart_quiz()` → gọi `controller.restart_quiz()`
    - `check_answer()` → gọi `controller.check_answer()`
    - `next_question()` → gọi `controller.next_question()`

Keyword / Kiến thức cần:

- `ScreenManager` , `FadeTransition`
- Cách khởi tạo nhiều Screen và đặt `name` cho từng màn hình

💡 **Gợi ý**:

- `name` của mỗi Screen phải khớp với tên dùng trong `sm.current = "..."`
- `FadeTransition` tạo hiệu ứng mờ dần khi chuyển màn hình
- `self.quiz_controller` nên được lưu trong App để các Screen gọi được

📝 **Các bước thực hiện**:

1. Tạo file `main.py` ở thư mục gốc
2. Import tất cả thành phần:

```python
from kivy.app import App
from kivy.uix.screenmanager import ScreenManager, FadeTransition
from models.quiz_model import QuizModel
from views.start_screen import StartScreen
from views.quiz_screen import QuizScreen
from views.result_screen import ResultScreen
from controllers.quiz_controller import QuizController
```

3. Tạo class `QuizApp(App)` :

```python
class QuizApp(App):
    def build(self):
```

4. Tạo ScreenManager:

```python
sm = ScreenManager(transition=FadeTransition())
```

5. Tạo 3 màn hình và thêm vào ScreenManager:

```python
start_screen = StartScreen(name="start")
quiz_screen = QuizScreen(name="quiz")
result_screen = ResultScreen(name="result")
sm.add_widget(start_screen)
sm.add_widget(quiz_screen)
sm.add_widget(result_screen)
```

6. Tạo Model và Controller:

```python
model = QuizModel(total_questions=5)
self.quiz_controller = QuizController(
    model=model,
    screen_manager=sm,
    quiz_screen=quiz_screen,
    result_screen=result_screen
)
```

7. Viết các hàm wrapper để View gọi:

```python
def start_quiz(self):
    self.quiz_controller.start_quiz()

def restart_quiz(self):
    self.quiz_controller.restart_quiz()

def check_answer(self, choice_index):
    return self.quiz_controller.check_answer(choice_index)

def next_question(self):
    self.quiz_controller.next_question()
```

8. Return ScreenManager:

```python
return sm
```

9. Thêm phần chạy app:

```python
if __name__ == "__main__":
    QuizApp().run()
```

10. Chạy thử: `python main.py` và test đầy đủ tất cả tính năng!

Task 11 (tuỳ chọn): Chỉnh màu sắc / giao diện

Mục tiêu: Cho học sinh tự sáng tạo UI.

Ý tưởng:

- Cho học sinh tự chọn bảng màu (tươi sáng / tối / theo chủ đề).
- Thay đổi các hằng số màu trong `views/widgets.py`.
- Thử tăng/giảm `font_size`, `padding`, `spacing`.

Keyword / Kiến thức cần:

- Hệ màu RGBA trong Kivy: `(R, G, B, A)` với giá trị 0–1
- Thuộc tính `font_size`, `padding`, `spacing`, `size_hint`

💡 **Gợi ý**:

- Dùng công cụ chọn màu online (Color Picker) để lấy mã màu RGB
- Chia giá trị RGB cho 255 để chuyển sang hệ 0-1 của Kivy
- Ví dụ: màu #3498db (52, 152, 219) → (0.2, 0.6, 0.86, 1)
- Có thể thêm hình nền bằng `canvas.before`

📝 **Các bước thực hiện:**

1. Mở file `views/widgets.py`
2. Tìm phần định nghĩa màu (ví dụ):

```python
PRIMARY_COLOR = (0.2, 0.6, 1, 1)
SECONDARY_COLOR = (0.95, 0.95, 0.95, 1)
```

3. Chọn bảng màu mới (ví dụ chủ đề tối):

```python
PRIMARY_COLOR = (0.1, 0.7, 0.4, 1)  # Xanh lá mint
BACKGROUND_COLOR = (0.15, 0.15, 0.2, 1)  # Xám tối
TEXT_COLOR = (1, 1, 1, 1)  # Trắng
```

4. Thay đổi kích thước chữ cho phù hợp:

```python
# Trong make_title:
font_size="32sp"  # Tăng từ 28sp
```

5. Điều chỉnh khoảng cách:

```python
# Trong các Screen:
padding=30  # Tăng từ 20
spacing=20  # Tăng từ 15
```

6. (Nâng cao) Thêm hình nền gradient hoặc hình ảnh
7. Chạy thử và điều chỉnh cho đẹp mắt
8. So sánh trước/sau và chia sẻ với lớp!

---

4. Gợi ý cách giao bài / kiểm tra

- Mỗi **Task** có thể là 1 buổi học hoặc 1 bài tập về nhà nhỏ.
- Giáo viên có thể yêu cầu:
  - Chụp màn hình kết quả chạy app ở từng bước.
  - Nộp mã nguồn từng file sau mỗi task.
  - Viết 3–5 câu mô tả lại: "Hôm nay em học được gì?".

Nếu thầy/cô muốn, tôi có thể tạo thêm phiên bản **worksheet** (phiếu học tập) riêng cho từng Task để in ra cho học sinh điền.