Gavin Giordano

Professor Richards

COMPSCI 326

17 December 2025

KGD

[KG1] Endpoint Definitions

app/routes.py

```
@router.get("/dashboard", response_class=HTMLResponse)
async def dashboard(request: Request, current_user: User =
Depends(auth.get_current_user_or_redirect), session: Session = Depends(get_session)):
```

This endpoint defines a unique URL path that a server exposes to clients for them to access. This fulfills the knowledge goal as it defines where resources can be accessed.

[KG2] HTTP Methods & Status Codes

app/routes.py

```
    response = RedirectResponse("/dashboard", status_code=303)
```

The status code here is 303, which in this case, has us being redirected from the login/signup page to the dashboard page. This shows the use of HTTP Methods and status codes to specify the action that is being performed on a resource.

[KG3] Endpoint Validation

app/models.py

```
class Entry(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    mood_score: int = Field(ge=1, le=5)
    comment: Optional[str] = Field(default=None)
    created_at: datetime = Field(
        default_factory=lambda: datetime.now(timezone.utc))
    user_id: int = Field(foreign_key="user.id")
```

The code here is a model for an entry. For the mood_score, it is validating that the mood is an integer from just the range 1-5. FastAPI will automatically perform endpoint validation with it.

[KG4] Dependency Injection

app/auth.py

```
def get_current_user(request: Request, session: Session = Depends(get_session)) ->
User:
```

Here, dependency injection is being used to get the current database session. This shows dependency injection as the get_session function is defined outside of the function, and we are injecting it into the get_current_user one without rewriting it.

[KG5] Data Model

app/models.py

```
class User(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    name: str
    username: str = Field(index=True, unique=True)
    hashed_password: str
```

This code snippet is a data model for the user, which defines data types like the name, which is a string, the username, which is also a string, and a hashed password which is also a string. It is a clear example of a data model showing how the data is stores in the database.

[KG6] CRUD Operations & Persistent Data

app/routes.py

```
    new_entry = Entry(mood_score=mood_score, comment=comment,
                      user_id=current_user.id)
    session.add(new_entry)
    session.commit()
    session.refresh(new_entry)
```

Here, a new entry is being created, and it is being added to the database. This is an example of the Create CRUD operation. It is also an example of persistent data as that entry is being stores in the PostgreSQL database.

[KG7] API Endpoints & JSON

app/routes.py

```
@router.get("/api/entries")
def api_list_entries(session: Session = Depends(get_session), current_user: User =
Depends(auth.get_current_user)):
    entries = session.exec(select(Entry).where(
```

```
        Entry.user_id == current_user.id)).all()
    entries_data = [
        {
            "id": entry.id,
            "mood_score": entry.mood_score,
            "comment": entry.comment,
            "created_at": entry.created_at.isoformat()
        } for entry in entries
    ]
    return JSONResponse(content={"entries": entries_data})
```

This endpoint is returning a list of all the entries of a user in JSON format. When you go to the endpoint, you will see all your past entries in JSON format.

[KG8] UI Endpoints & HTMX

app/templates/dashboard.html

```
        <div id="entries-list" hx-get="/entries" hx-trigger="load" hx-
swap="innerHTML"></div>
```

Here, I'm using HTMX to request and render HTML from the /entries endpoint. It updates part of the page without a full reload.

[KG9] User Interaction (CRUD)

app/templates/fragments/entry.html

```
            <button hx-get="/entries/{{entry.id}}/edit" hx-target="#entry-
{{entry.id}}" hx-swap="outerHTML">
```

When you click on this button, you are able to edit a past entry, which shows the CRUD operating from the user's perspective.

[KG10] Separation of Concerns

templates/

routes.py

models.py

database.py

This is the main file structure of my code. I am separating the major components of the web app into different files, those being the front-end html files, my FastAPI endpoints, my data models, and my database, respectively.