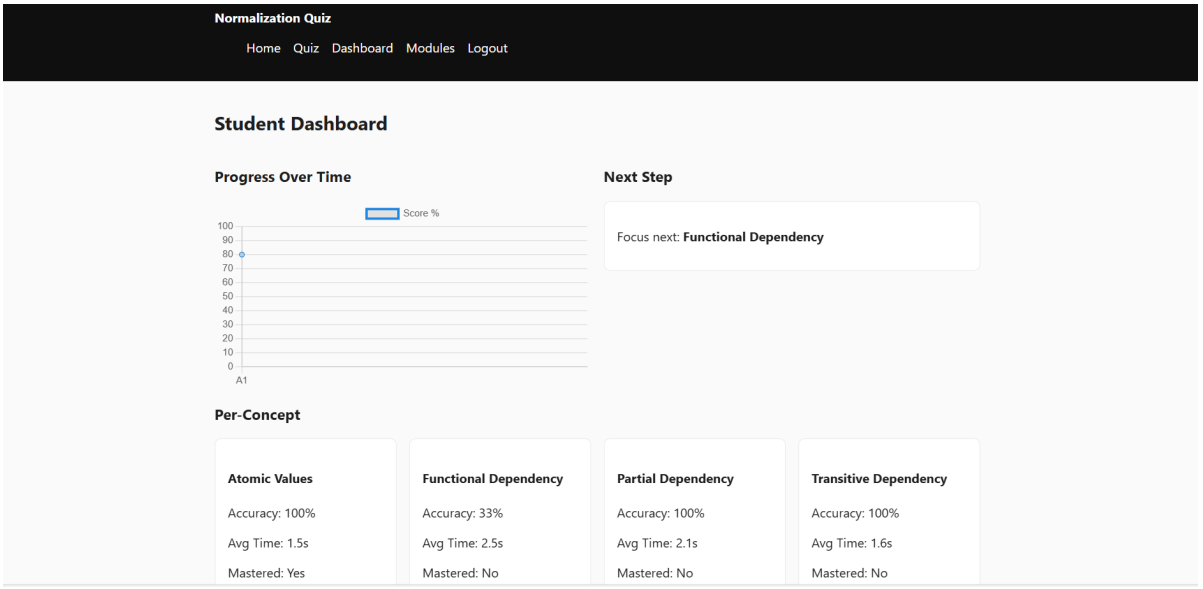
Current Progress 14/10

Dashboard (Student's View)

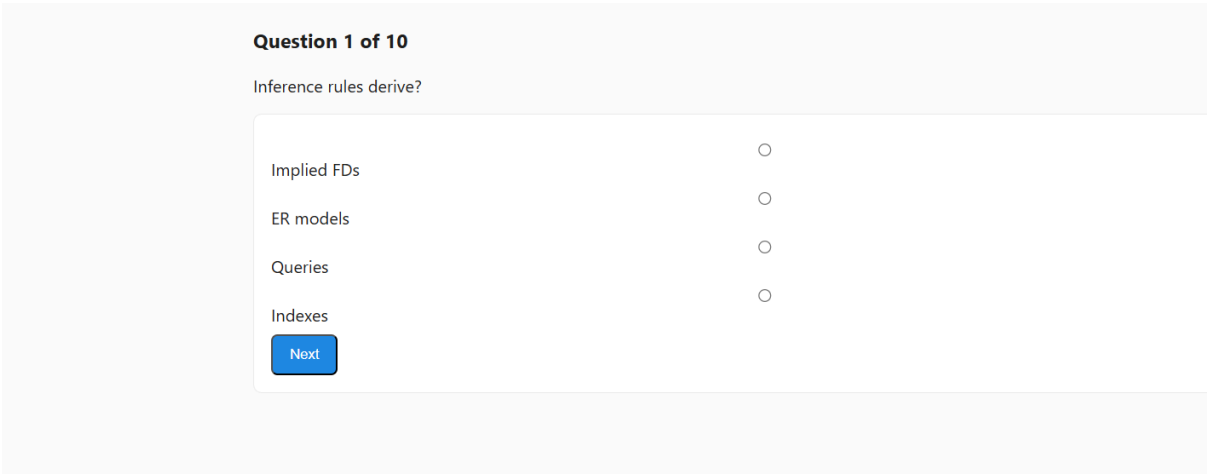


<u>Features</u>

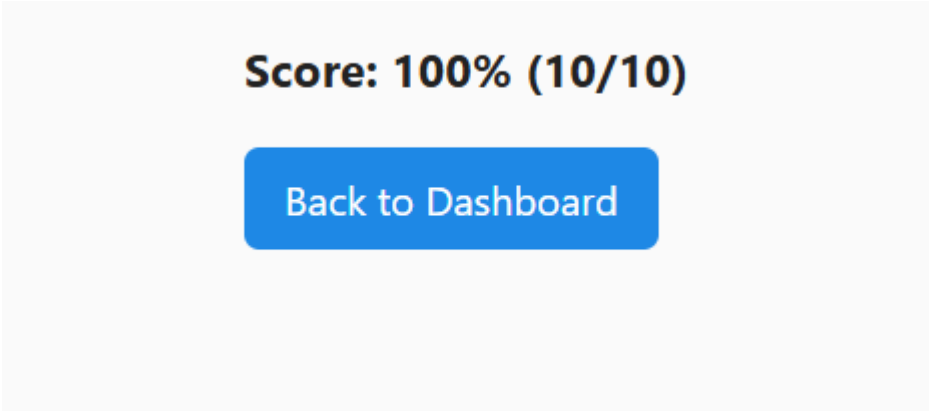
- Student recent score test results
- Student's next step focus
- Student's mastering level per concept
- Last attempt details (all questions and chosen answer with the correct or false) + time taken(+ category(fast, normal, slow)

Main Menu

- Home
- Quiz

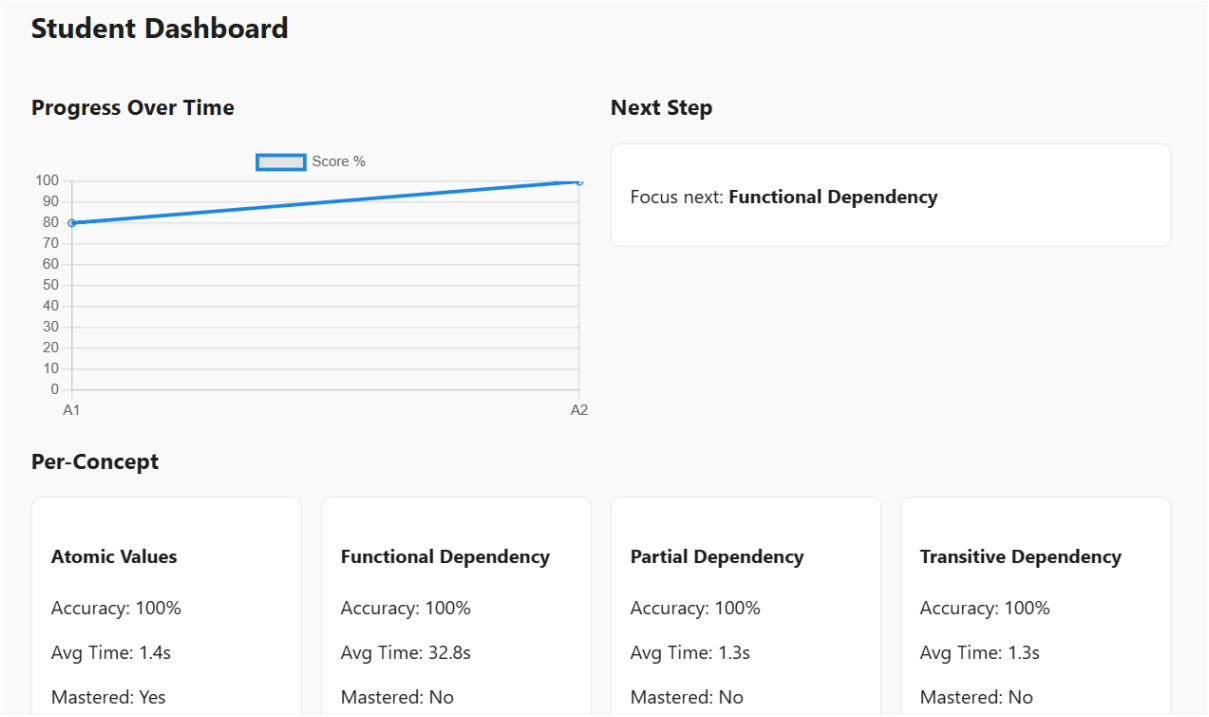


Quiz so basic, total 10 questions

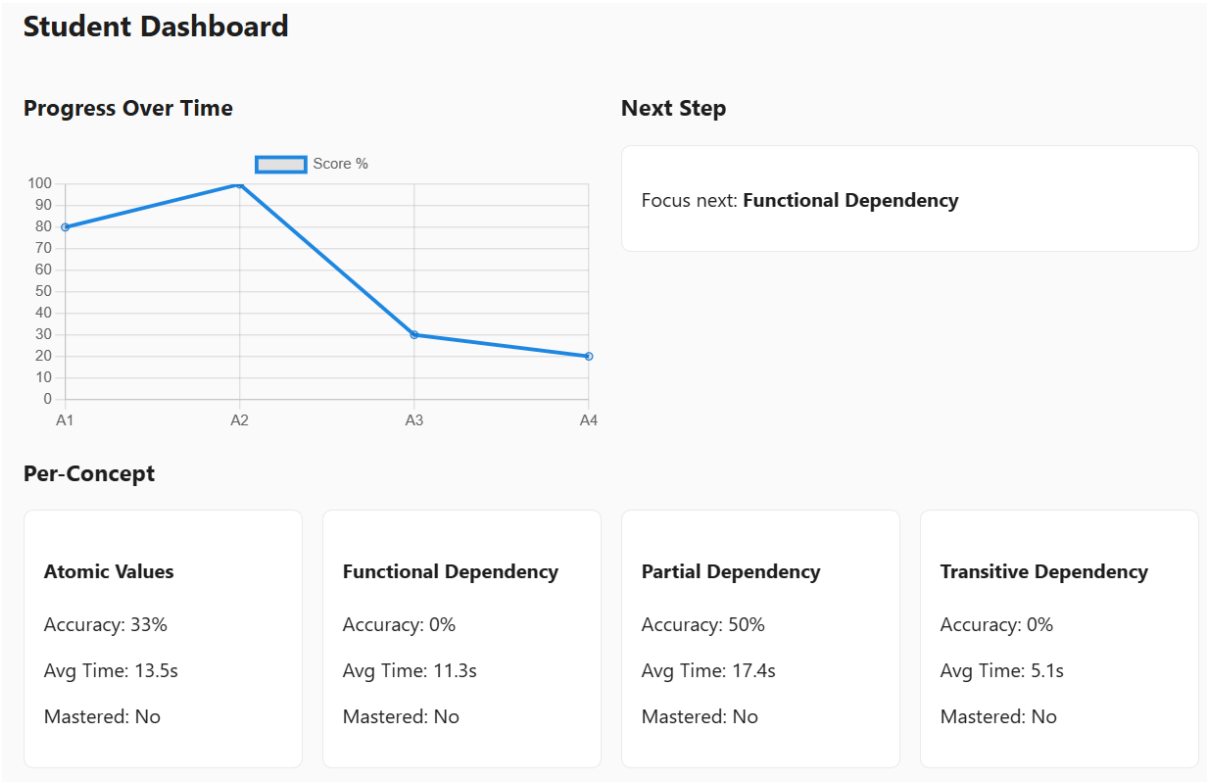


After done only show score.

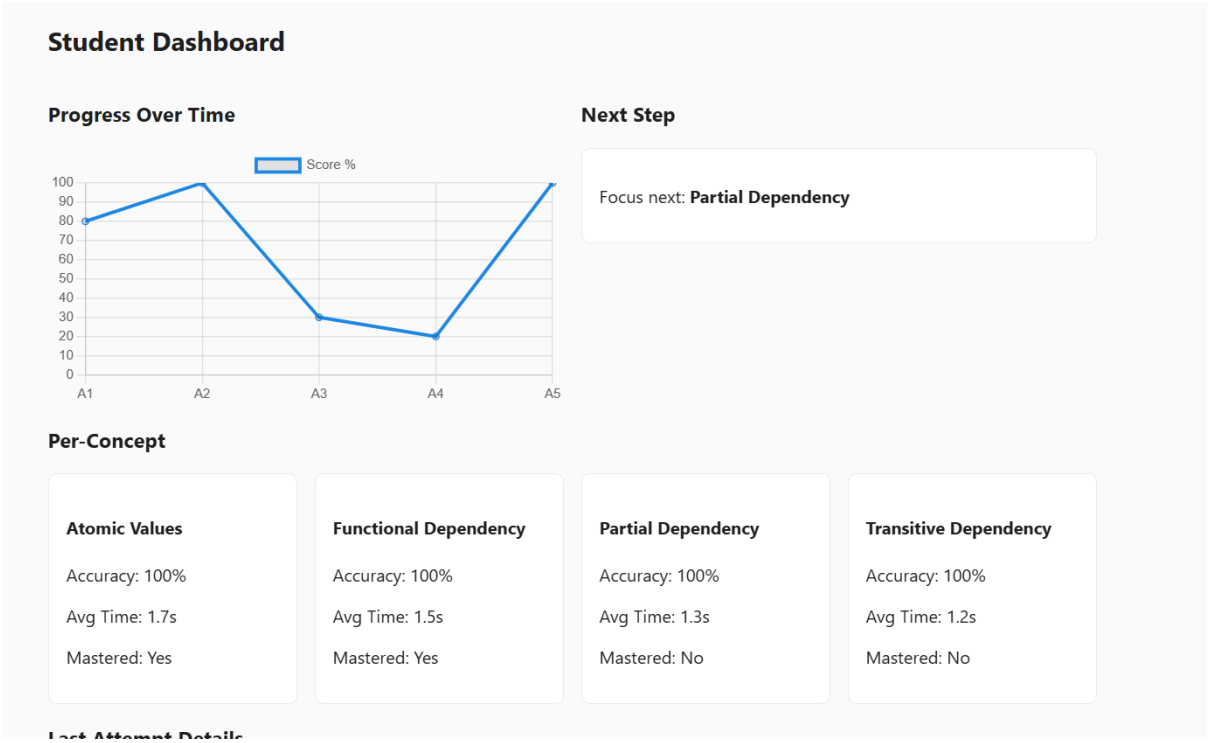
- Dashboard



- After 2nd attempt



After 4th attempt



After 5Th Attempt

- Modules

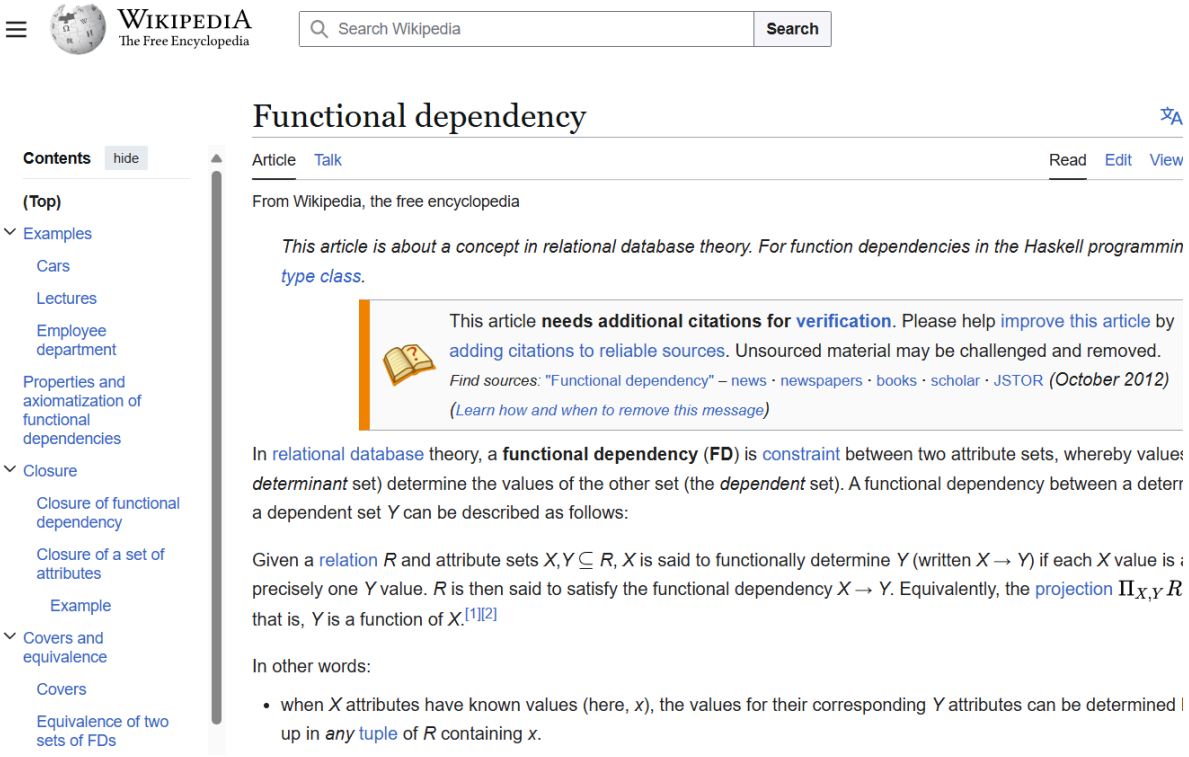
Functional Dependencies 101

Level: FD | **Concept:** Functional Dependency

Understand determinants and implied attributes

Open Resource

-



WIKIPEDIA
The Free Encyclopedia

Search Wikipedia | Search

Functional dependency

Article | Talk | Read | Edit | View

From Wikipedia, the free encyclopedia

This article is about a concept in relational database theory. For function dependencies in the Haskell programmin
type class.

This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.
Find sources: "Functional dependency" – news · newspapers · books · scholar · JSTOR *(October 2012)*
(Learn how and when to remove this message)

In relational database theory, a **functional dependency** (**FD**) is constraint between two attribute sets, whereby value
determinant set) determine the values of the other set (the *dependent* set). A functional dependency between a deter
a dependent set *Y* can be described as follows:

Given a relation *R* and attribute sets $X, Y \subseteq R$, *X* is said to functionally determine *Y* (written $X \rightarrow Y$) if each *X* value is
precisely one *Y* value. *R* is then said to satisfy the functional dependency $X \rightarrow Y$. Equivalently, the projection $\Pi_{X,Y} R$
that is, *Y* is a function of *X*.[1][2]

In other words:

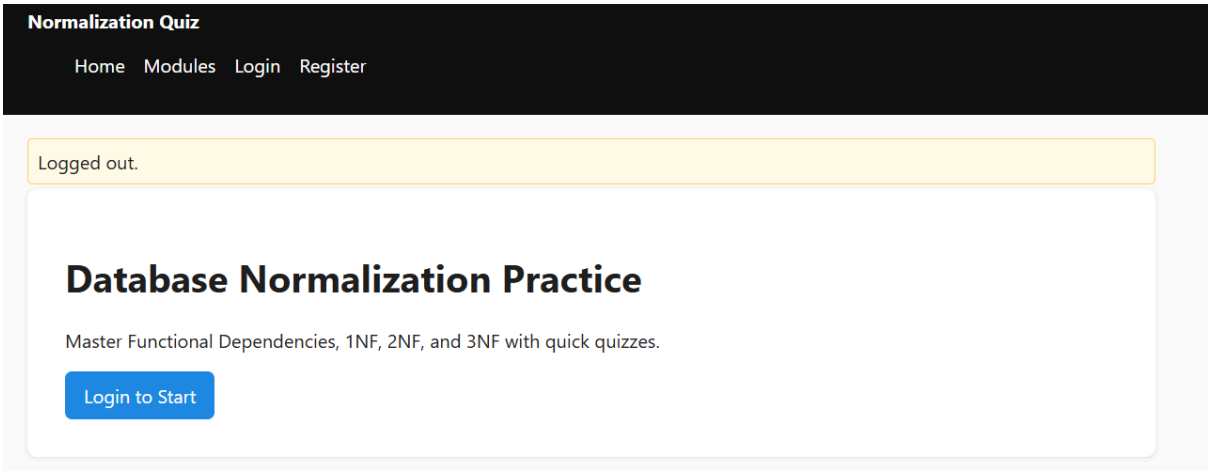
- when *X* attributes have known values (here, *x*), the values for their corresponding *Y* attributes can be determined
up in *any* tuple of *R* containing *x*.

Contents hide

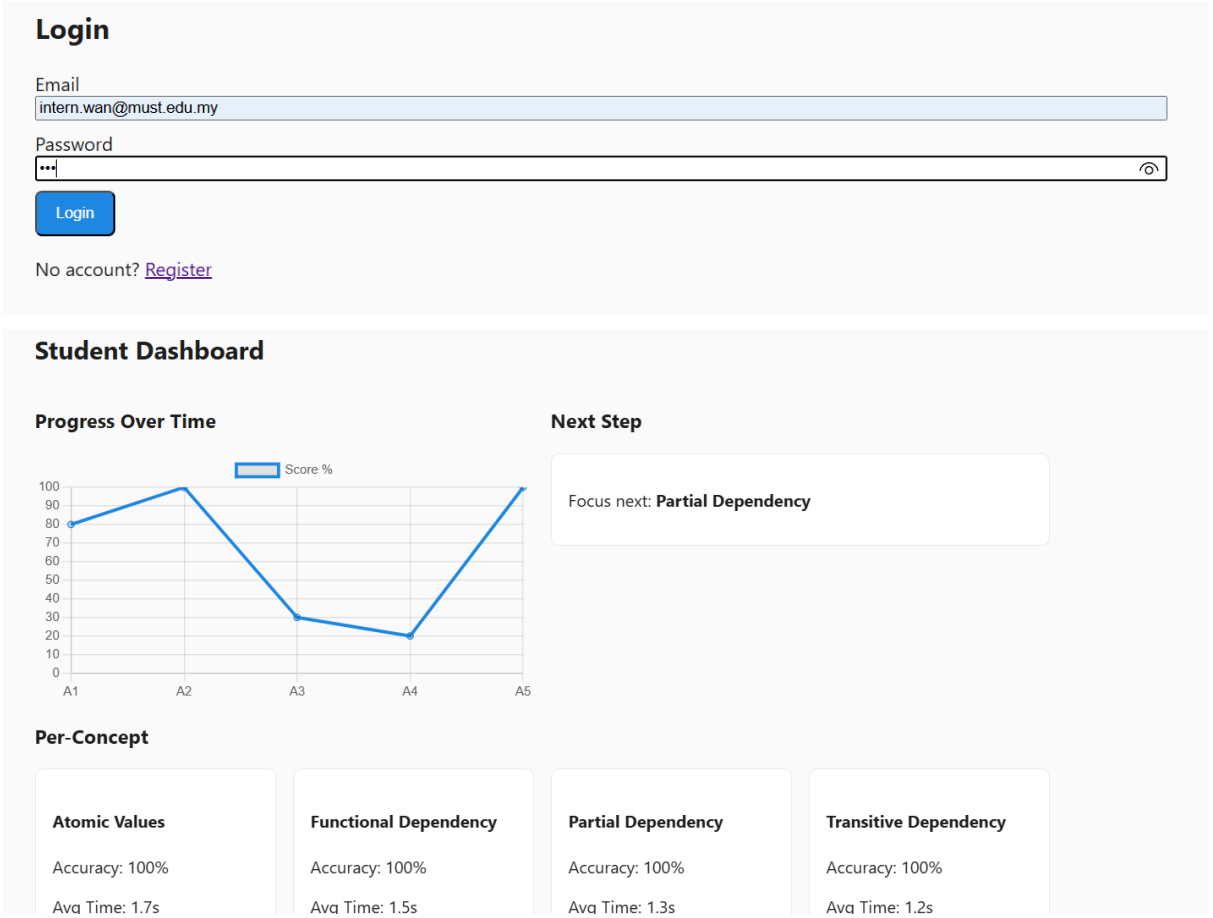
(Top)
- Examples
  - Cars
  - Lectures
  - Employee department
- Properties and axiomatization of functional dependencies
- Closure
  - Closure of functional dependency
  - Closure of a set of attributes
    - Example
- Covers and equivalence
  - Covers
  - Equivalence of two sets of FDs

-

- Logout



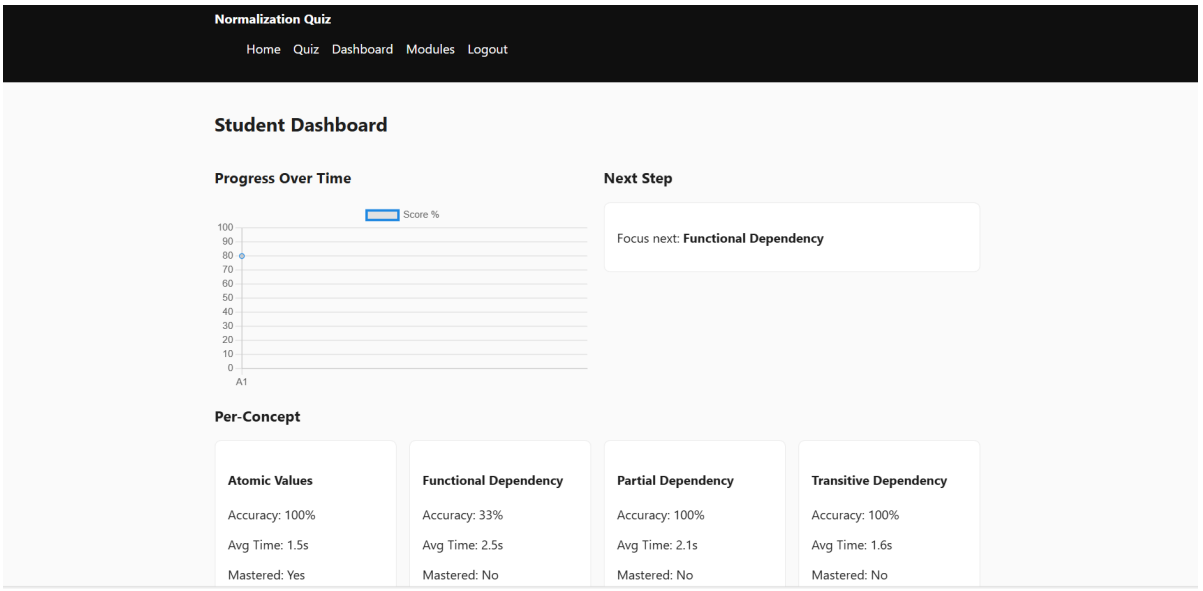
2nd Session



PROGRESS SAVED !!!

Issues

Dashboard (Student's View)

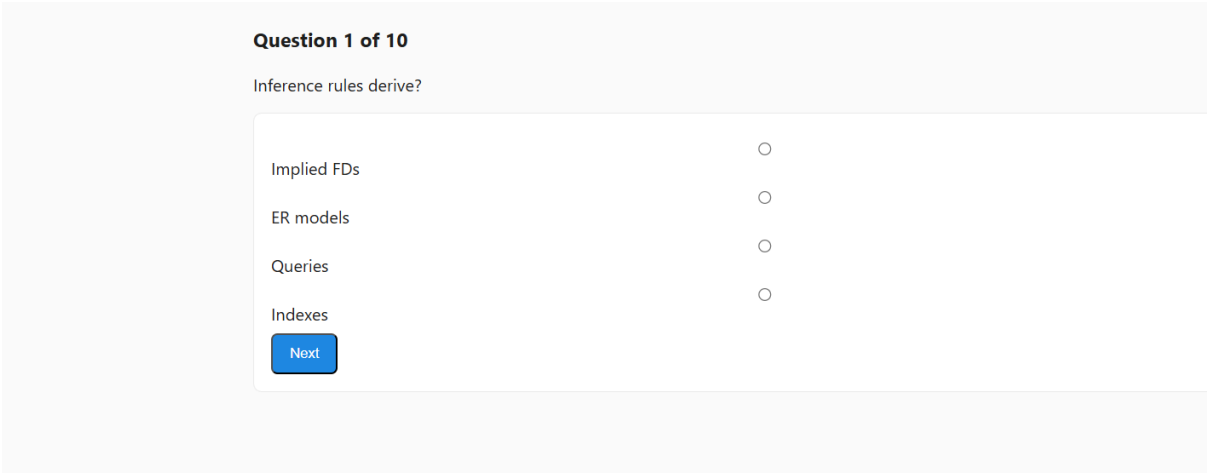


<u>Features</u>

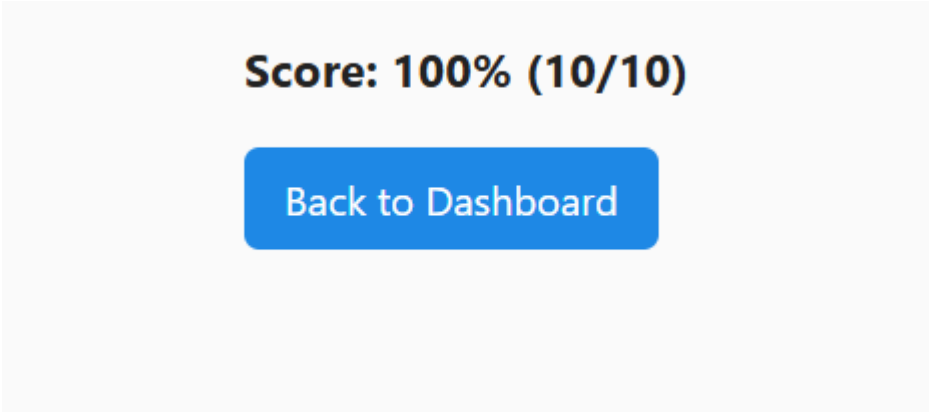
- Student recent score test results - **(Can put more better UI with reattempt button)**
- Student's next step focus - **(Can put the in depth topic focus to achieve next level with percentage gauge)**
- Student's mastering level per concept **(Good function but more questions can justify the purpose. + No need time taken for the student view, only accuracy and mastered.**
- Last attempt details (all questions and chosen answer with the correct or false) + time taken(+ category(fast, normal, slow)- **(Can only focus on the topic rather than the question and put whether mastered and add next topic for more understanding of succession.**

Main Menu

- Home
- Quiz

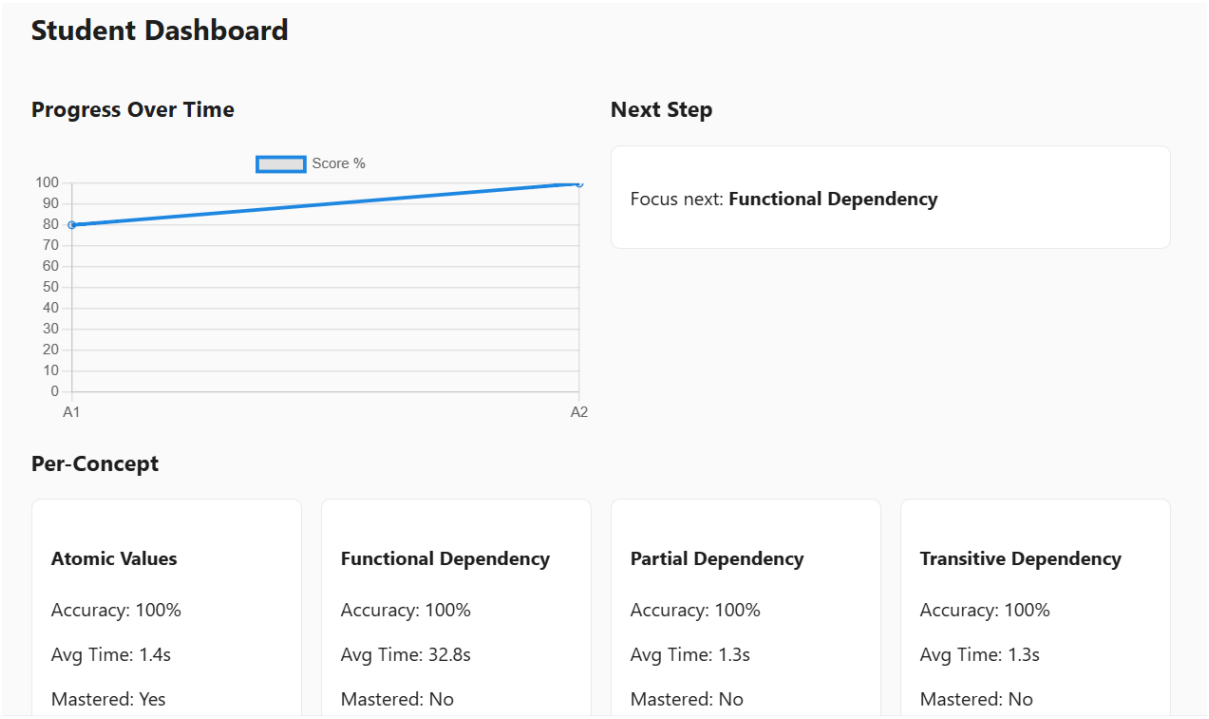


The interface looks so basic (We can replicate CanvasLMS or moodle interface)

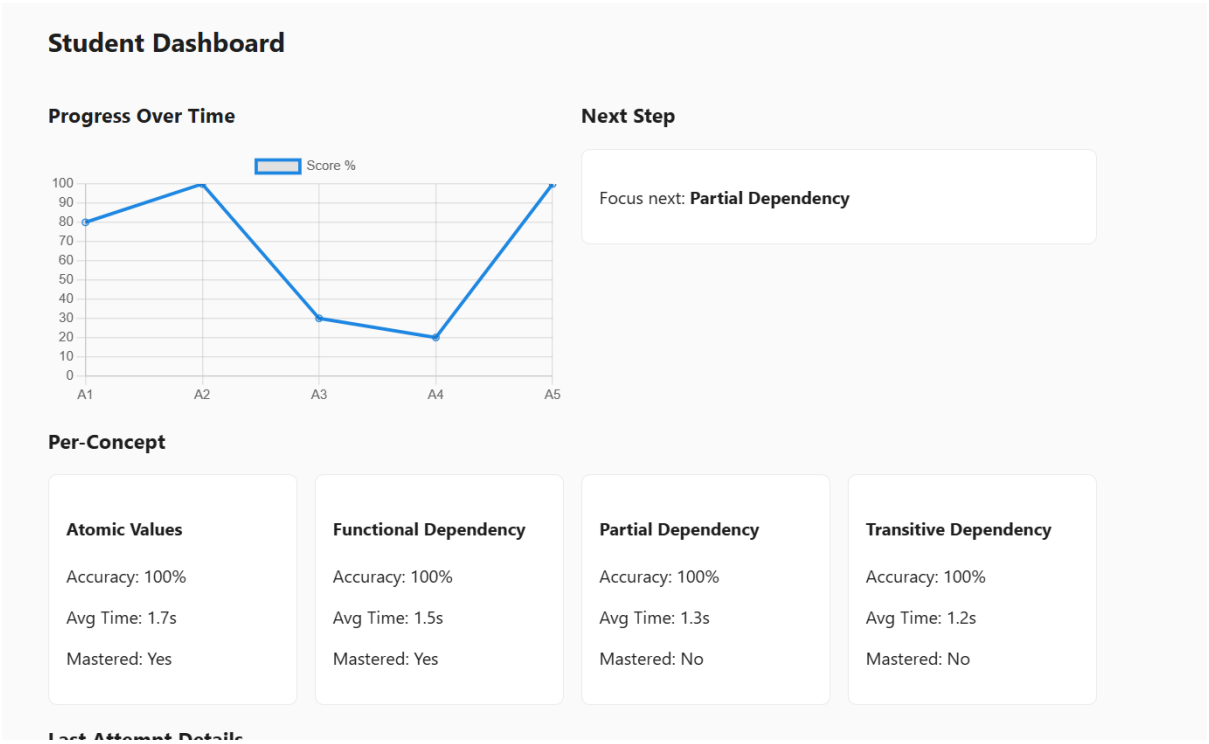


After done only show score. (**Just show the percentage). Add more questions. - Follow the data by forms results.**
If fail- Show retake quiz
If above 70 , show next topic to proceed

- Dashboard

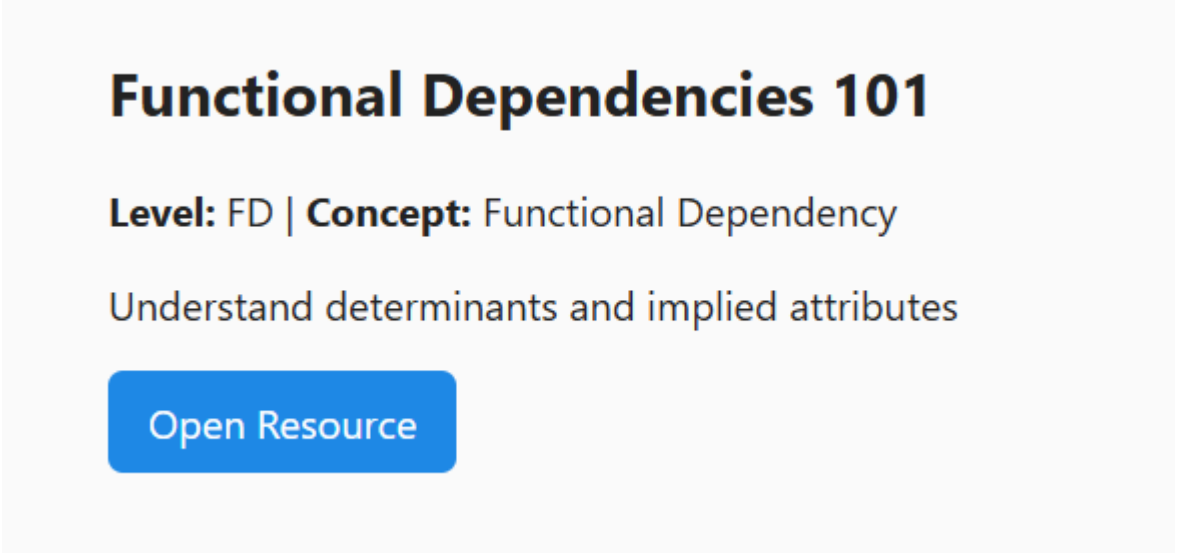


- After 2nd attempt - (See how the each question timing is calculated - for lecturer dashboard reference).



After 5Th Attempt (The Progress is not detailed, after 5 attempt. The student should have a grasp of their progress at this point.)

- Modules



-
- Add resource from the text book according to the topic

Chapter 12 Normalization

ADRIENNE WATT

Normalization should be part of the database design process. However, it is difficult to separate the normalization process from the ER modelling process so the two techniques should be used concurrently.

Use an entity relation diagram (ERD) to provide the big picture, or macro view, of an organization's data requirements and operations. This is created through an iterative process that involves identifying relevant entities, their attributes and their relationships.

Normalization procedure focuses on characteristics of specific entities and represents the micro view of entities within the ERD.

What Is Normalization?

Normalization is the branch of relational theory that provides design insights. It is the process of determining how much redundancy exists in a table. The goals of normalization are to:

- Be able to characterize the level of redundancy in a relational schema
- Provide mechanisms for transforming schemas in order to remove redundancy

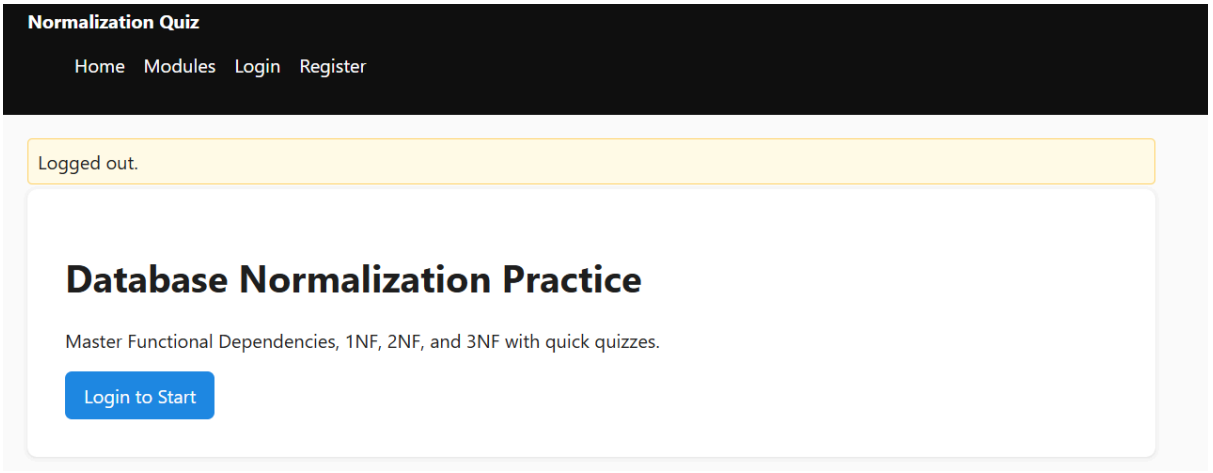
Normalization theory draws heavily on the theory of functional dependencies. Normalization theory defines six normal forms (NF). Each normal form involves a set of dependency properties that a schema must satisfy and each normal form gives guarantees about the presence and/or absence of update anomalies. This means that higher normal forms have less redundancy, and as a result, fewer update problems.

Normal Forms

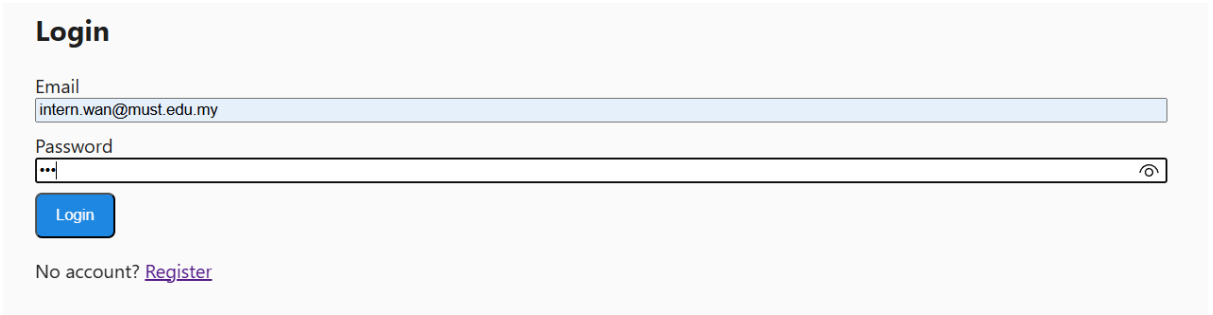
All the tables in any database can be in one of the normal forms we will discuss next. Ideally we only want minimal redundancy for PK to FK. Everything else should be derived from other tables. There are six normal forms, but we will only look at the first four, which are:

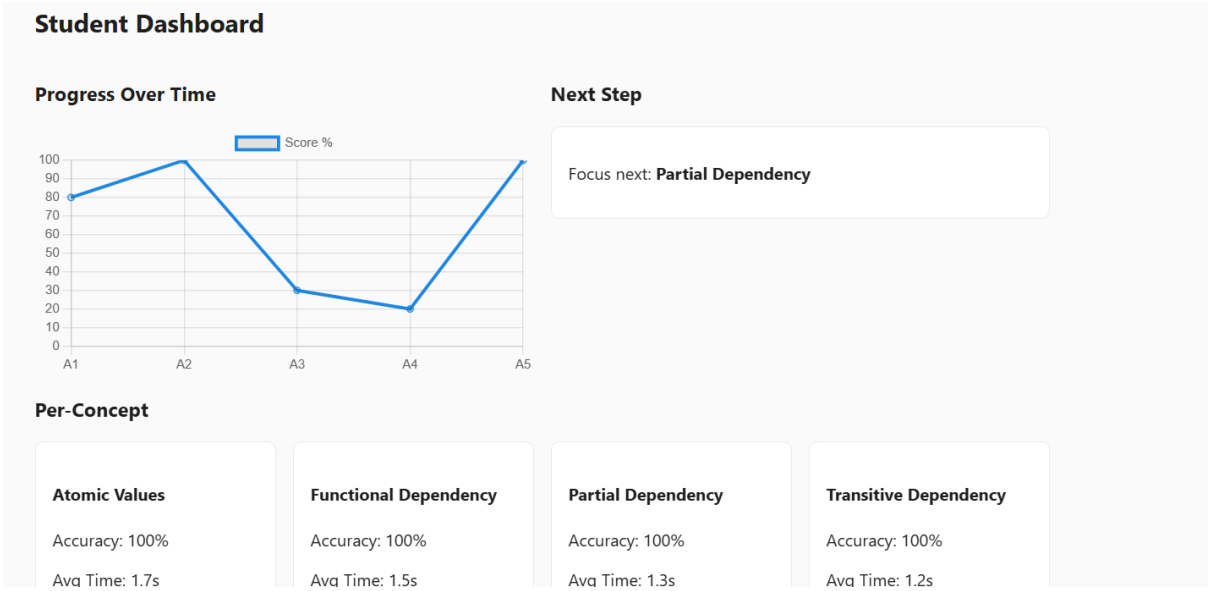
-

- Logout **(Add logout at the side of the page, separately)**



2nd Session(On the login or register page, dont add the menu. Just login or register)





PROGRESS SAVED !!!

—----------

🚀 Coding Roadmap

Phase 1: Student Side (highest priority)

1. **Detailed Feedback (Task 1)**
    - Add *Your Answer, Correct Answer, Explanation* after each question submission.
    - Reason: Core learning feature, builds trust.
2. **Progress Tracking Visuals (Task 2)**
    - Add per-concept progress bars and a “Completion Badge”.
    - Reason: Makes progress measurable and motivating.
3. **“Next Step” Card (Task 3)**
    - Highlight the most recent weak concept on dashboard.
    - Reason: Ensures students know exactly what to do next.
4. **Polish & Error Handling (Task 4)**
    - Friendly empty-state messages, confirmation banners.
    - Reason: Improves usability and avoids confusion.
5. **Student Profile Header (Task 5, optional polish)**
    - Add student’s name/program to dashboard header.
    - Reason: Personalization, makes demo more professional.

Phase 2: Educator Side

1. **Class Summary Charts polish (Task 1)**
    - Labels on bars, show % mastered.
    - Reason: Gives quick overview of class readiness.
2. **Student-by-Concept Table enhancements (Task 2)**
    - Add mastery status badges (Mastered / In progress).
    - Reason: Makes weak students easy to spot.
3. **Recommendation Tracking (Task 3)**
    - Add “Pending / Reassessed” column.
    - Reason: Lets lecturer see if students acted on feedback.
4. **Student Drill-Down (Task 4)**
    - Click student name → go to that student’s dashboard.
    - Reason: Deep dive into individuals.
5. **Export to CSV (Optional Task 5)**
    - Export class summary, student performance, recommendations.
    - Reason: Useful for lecturers who want offline reports.

✅ Summary

- **Phase 1 (Student Side):** 5 tasks → must finish before demo.
- **Phase 2 (Educator Side):** 5 tasks → builds on Phase 1 data, less urgent but good for completene