IIT Madras

BSc Degree

Controllers

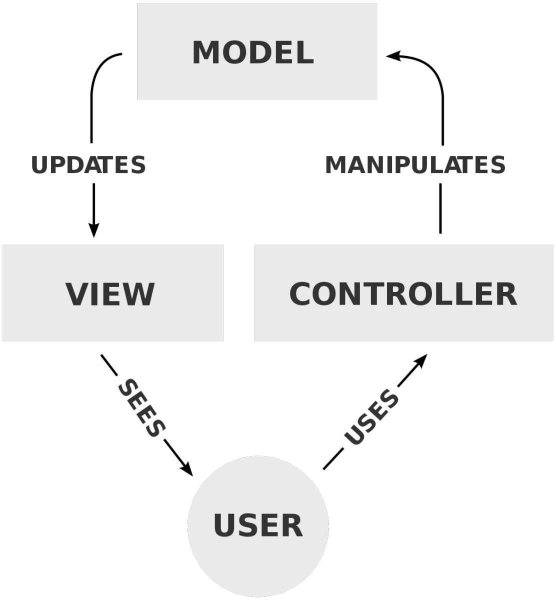
Taking Action

- Origins of MVC
- Request - Response
- Group Actions: Controller
- CRUD
- Routes and Controllers

MVC Origins

Model-View-Controller

- Design pattern - or collection of design patterns
- Originally introduced in context of GUI design in Smalltalk-80
- Many different variants, interpretations...



By RegisFrey - Own work, Public Domain, Wikipedia

From: Trygve Reenskaug

Date: 10 December 1979

MODELS - VIEWS - CONTROLLERS

MODELS

Models represent knowledge. A model could be a single object (rather uninteresting), or it could be some structure of objects. ~~The proposed implementation supports knowledge represented in something resembling *semantic nets* (If I understand Laura correctly)~~

VIEWS

A view is a (visual) representation of its model. It would ordinarily highlight certain attributes of the model and suppress others. It is thus acting as a *presentation filter*.

A view is attached to its model (or model part) and gets the data necessary for the presentation from the model by asking questions. It may also update the model by sending appropriate messages. All these questions and messages have to be in the terminology of the model, the view will therefore have to know the semantics of the attributes of the model it represents. (It may, for example, ask for the model's identifier and expect an instance of Text, it may not assume that the model is of class Text.)

CONTROLLERS

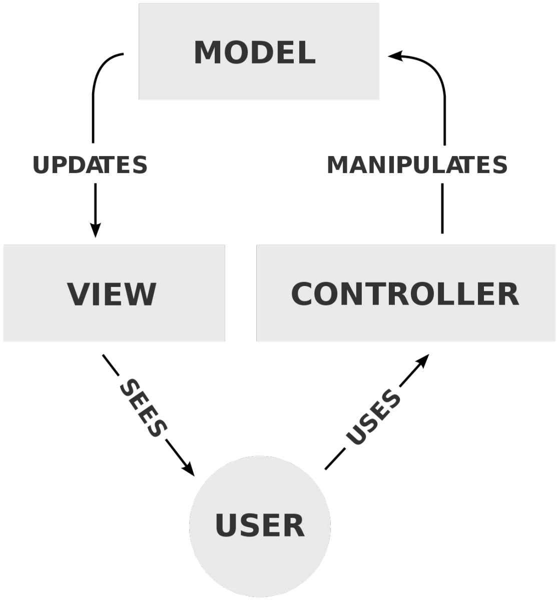
A controller is the link between a user and the system. It provides the user with input by arranging for relevant views to present themselves in appropriate places on the screen. It provides means for user output by presenting the user with menus or other means of giving commands and data. The controller receives such user output, translates it into the appropriate messages and pass these messages on .to one or more of the views.

A controller should never supplement the views, it should for example never connect the views of nodes by drawing arrows between them.

Conversely, a view should never know about user input, such as mouse operations and keystrokes. It should always be possible to write a method in a controller that sends messages to views which exactly reproduce any sequence of user commands.

General Concept - Action

- Take action in response to user input
- Communicate with the model, extract the view



Applicability

- Originally designed for GUI applications
- Separation of concerns - model vs view - and connection through controller
- State of interaction maintained as part of overall system memory

Applicability

- Originally designed for GUI applications
- Separation of concerns - model vs view - and connection through controller
- State of interaction maintained as part of overall system memory

Web?

- Server does not maintain state of client
- Client is pure front-end to user
- Some of the analogies break down - hence many variants of MVC

Present Context

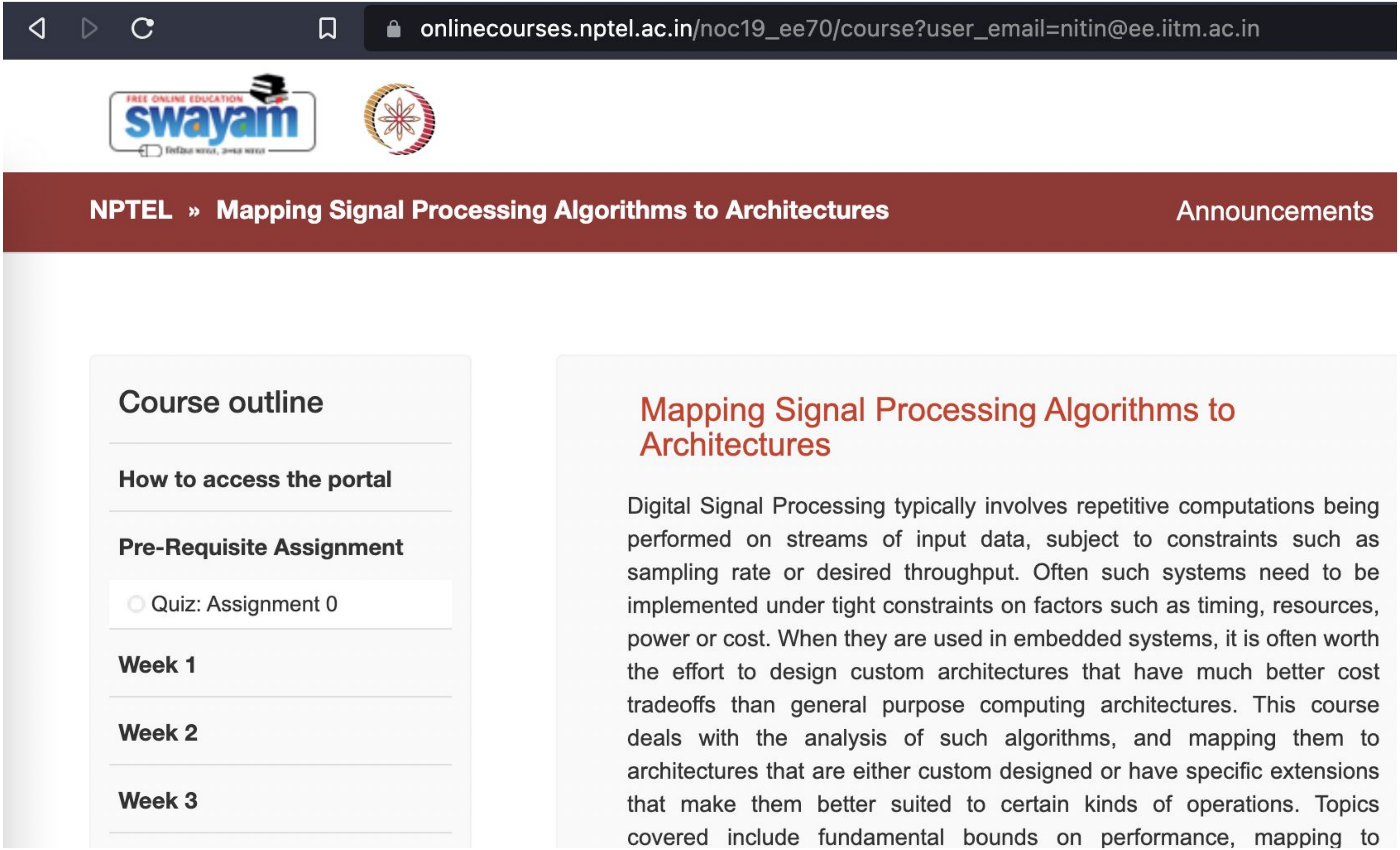
- MVC is a good conceptual framework to understand separation of concerns
- Breaks down if applied too rigidly
- The web in general does not have the close knit structure of GUI applications needed for MVC
- Other aspects like static typing and type inference of objects also broken in Python-like languages

Apply basic learnings from MVC, but be prepared to stretch

Requests and Responses

Example dynamic web page

- View: page
- Links: clickable to select various options
- Clicking a link **triggers** different behaviours



onlinecourses.nptel.ac.in/noc19_ee70/course?user_email=nitin@ee.iitm.ac.in
FREE ONLINE EDUCATION
swayam
NPTEL » Mapping Signal Processing Algorithms to Architectures
Announcements

Course outline

How to access the portal

Pre-Requisite Assignment

○ Quiz: Assignment 0

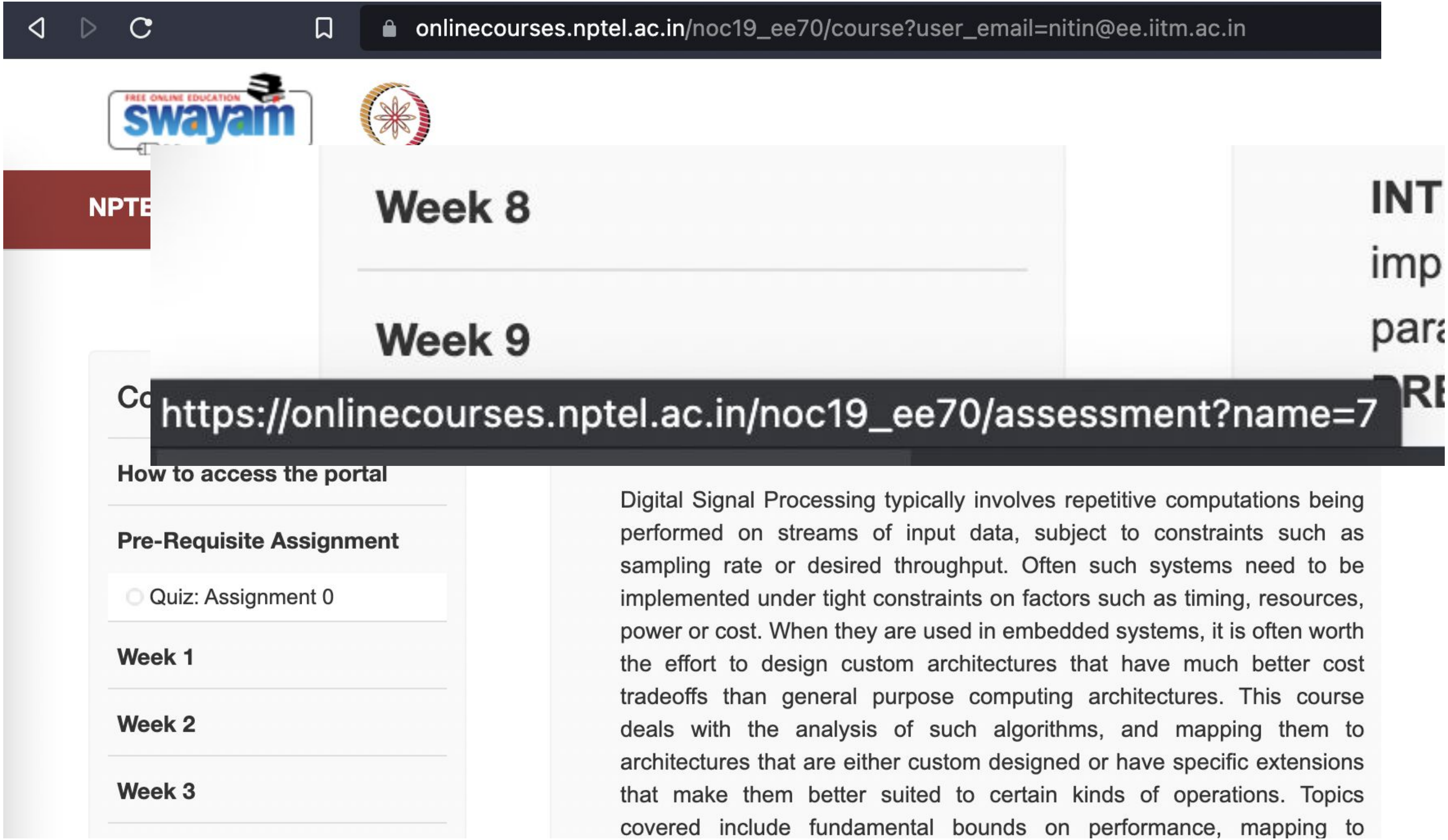
Week 1

Week 2

Week 3

Mapping Signal Processing Algorithms to Architectures

Digital Signal Processing typically involves repetitive computations being performed on streams of input data, subject to constraints such as sampling rate or desired throughput. Often such systems need to be implemented under tight constraints on factors such as timing, resources, power or cost. When they are used in embedded systems, it is often worth the effort to design custom architectures that have much better cost tradeoffs than general purpose computing architectures. This course deals with the analysis of such algorithms, and mapping them to architectures that are either custom designed or have specific extensions that make them better suited to certain kinds of operations. Topics covered include fundamental bounds on performance, mapping to



onlinecourses.nptel.ac.in/noc19_ee70/course?user_email=nitin@ee.iitm.ac.in

Week 8

Week 9

https://onlinecourses.nptel.ac.in/noc19_ee70/assessment?name=7

How to access the portal

Pre-Requisite Assignment

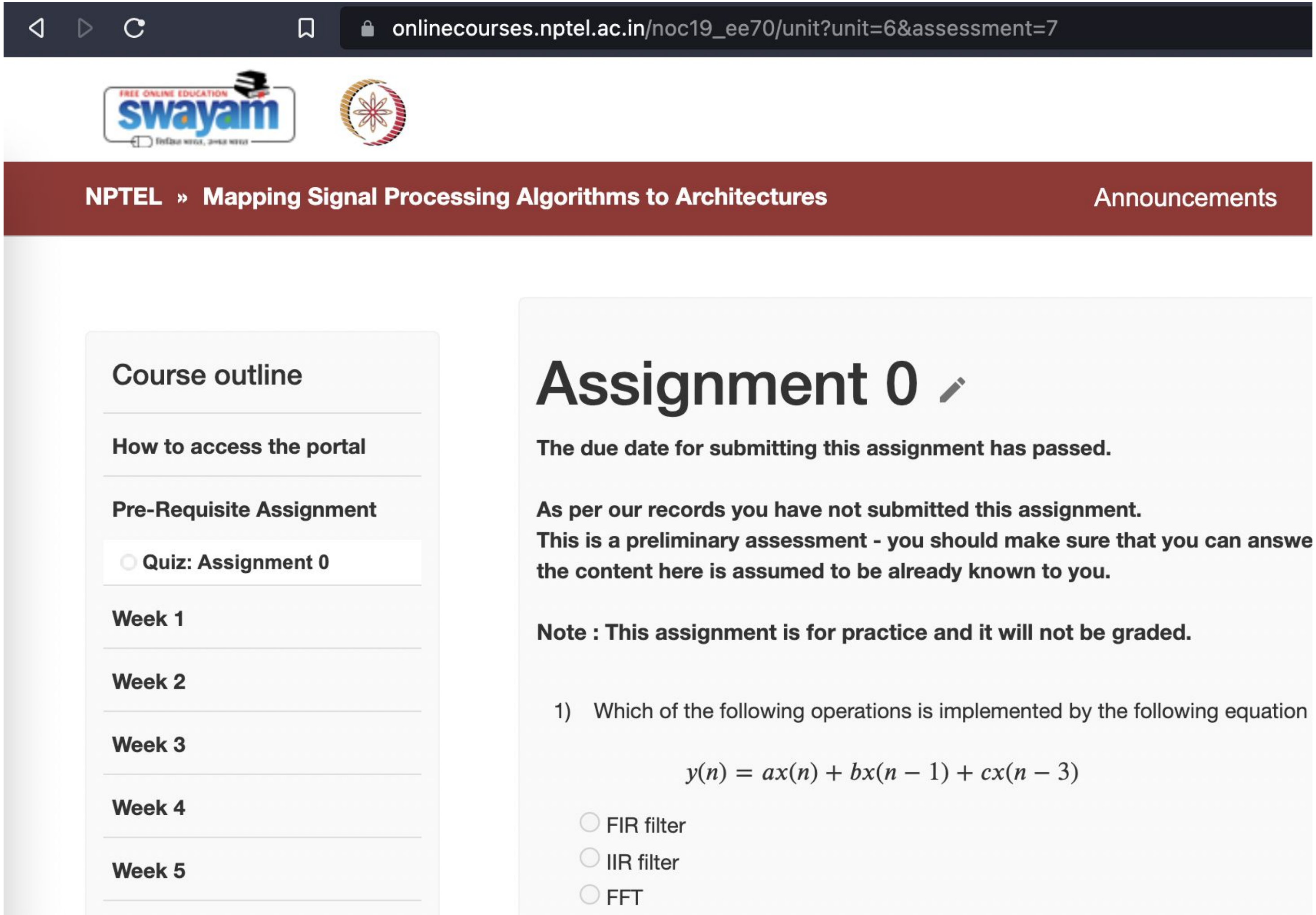
○ Quiz: Assignment 0

Week 1

Week 2

Week 3

Digital Signal Processing typically involves repetitive computations being performed on streams of input data, subject to constraints such as sampling rate or desired throughput. Often such systems need to be implemented under tight constraints on factors such as timing, resources, power or cost. When they are used in embedded systems, it is often worth the effort to design custom architectures that have much better cost tradeoffs than general purpose computing architectures. This course deals with the analysis of such algorithms, and mapping them to architectures that are either custom designed or have specific extensions that make them better suited to certain kinds of operations. Topics covered include fundamental bounds on performance, mapping to



Assignment 0

The due date for submitting this assignment has passed.

As per our records you have not submitted this assignment.
This is a preliminary assessment - you should make sure that you can answe
the content here is assumed to be already known to you.

Note : This assignment is for practice and it will not be graded.

1) Which of the following operations is implemented by the following equation

$$y(n) = ax(n) + bx(n-1) + cx(n-3)$$

○ FIR filter
○ IIR filter
○ FFT

Request - Response

- Web is based completely on requests and responses
  - Client makes requests
  - Server sends responses
- Basic requests: clicking on link / URL
  - HTTP GET
- More complex requests: *form* submissions
  - HTTP POST

Constraints?

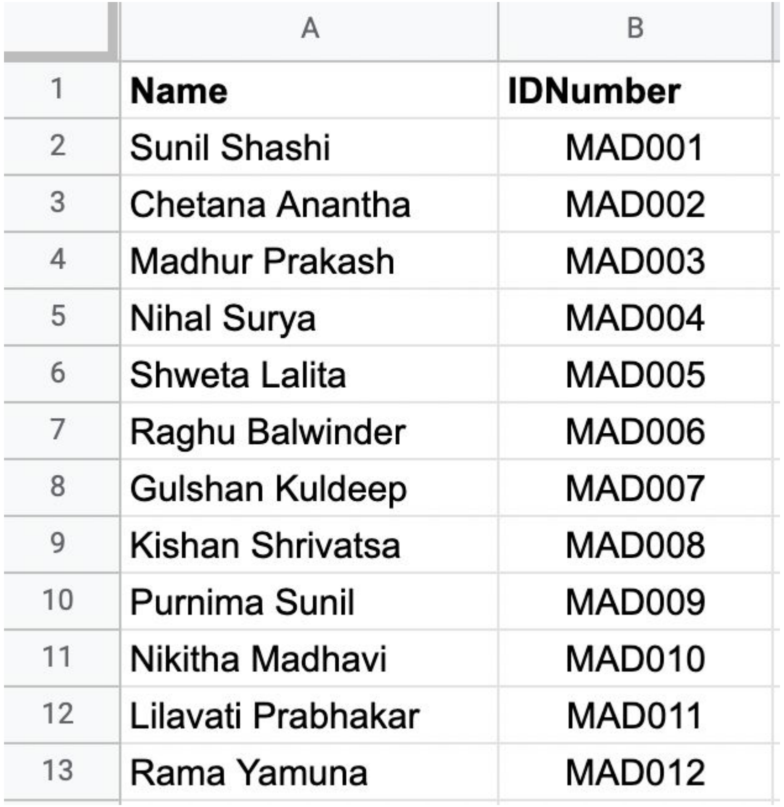
- Any “page” can be requested
- Assignments, quizzes, lectures, general information

Are there common threads?

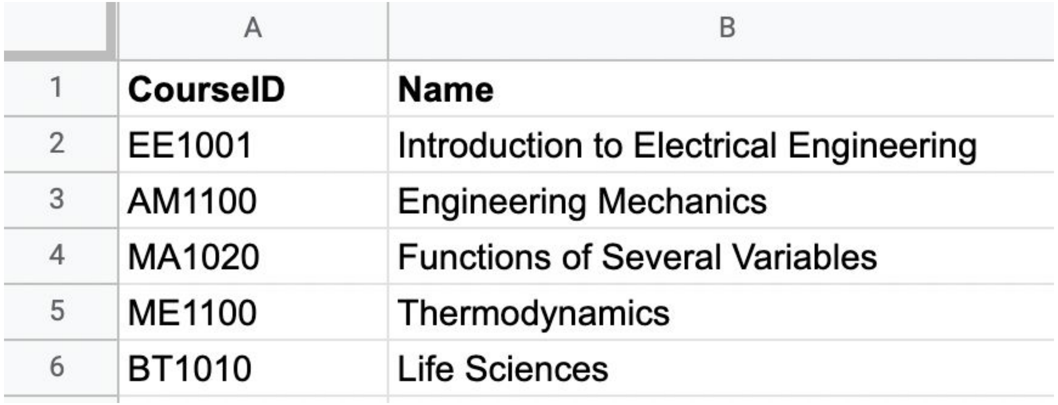
Example: Gradebook

- Students: ID, name, address, …
- Courses: ID, name, department, year, …
- StudentCourse Relationship: which students are registered for which courses

Example: Gradebook



	A	B
1	**Name**	**IDNumber**
2	Sunil Shashi	MAD001
3	Chetana Anantha	MAD002
4	Madhur Prakash	MAD003
5	Nihal Surya	MAD004
6	Shweta Lalita	MAD005
7	Raghu Balwinder	MAD006
8	Gulshan Kuldeep	MAD007
9	Kishan Shrivatsa	MAD008
10	Purnima Sunil	MAD009
11	Nikitha Madhavi	MAD010
12	Lilavati Prabhakar	MAD011
13	Rama Yamuna	MAD012



	A	B
1	**CourseID**	**Name**
2	EE1001	Introduction to Electrical Engineering
3	AM1100	Engineering Mechanics
4	MA1020	Functions of Several Variables
5	ME1100	Thermodynamics
6	BT1010	Life Sciences

Common Operations

- Create a new student - add name, roll number, date of birth, …
- Create a new course
- Assign student to course
- Enter marks for student / Update marks of student
- View summaries / charts / histograms
- Archive an old course
- Remove graduated students

Some essential functions can be distilled...