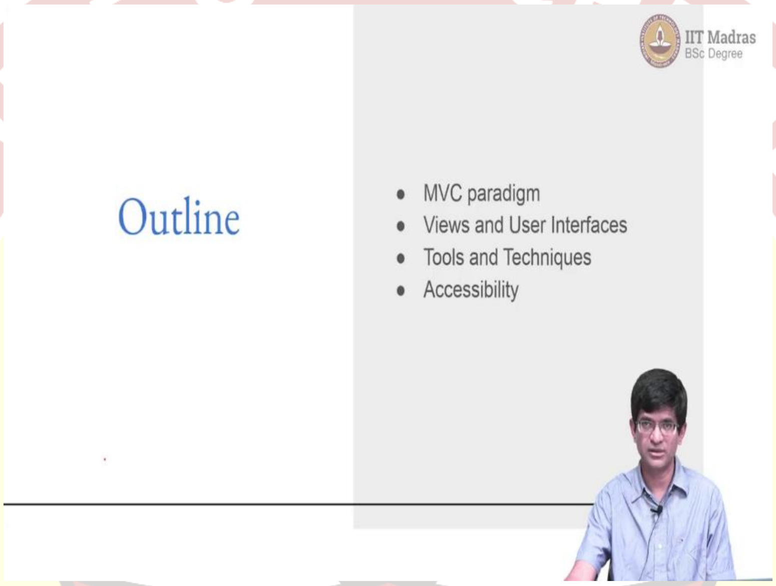
IIT Madras

ONLINE DEGREE

Modern Application Development - I
Professor. Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Overview of MVC

Hello, everyone, and welcome to this course on modern application development. Today, we will be talking about views in the context of web applications.

(Refer Slide Time: 0:24)



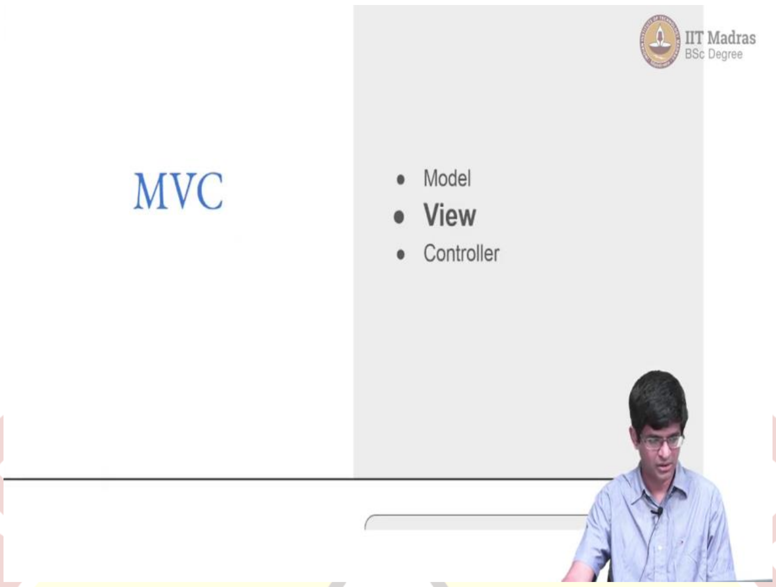
Outline

- MVC paradigm
- Views and User Interfaces
- Tools and Techniques
- Accessibility

So, the outline of the talk is roughly like this, I had already mentioned the model view controller paradigm, the MVC paradigm, we will briefly look at that, once again review that, then we will go a bit deeper into the V part of MVC, which is the views and user interfaces, I will then describe some tools and techniques that can be used for implementing and understanding views.

And one particular aspect of this that I feel needs to be emphasised is accessibility. And I will make a little bit more of a detour on that and go into a little bit more in depth on those on that part of the content.

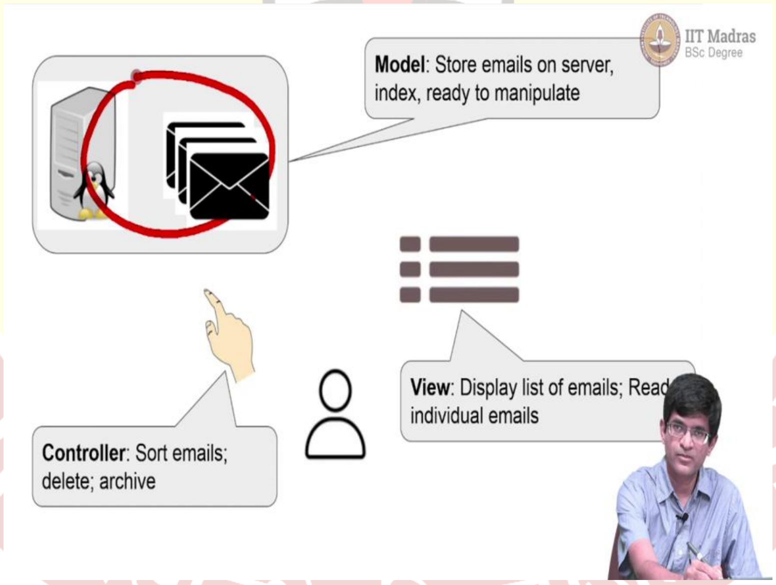
(Refer Slide Time: 1:03)


MVC

- Model
- **View**
- Controller

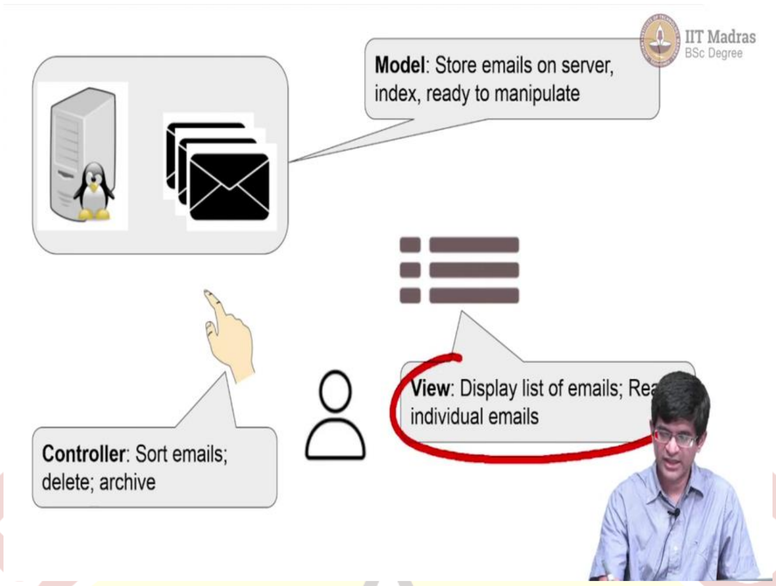
So, like I said, what we are doing over here is looking at the view part of the model view controller.

(Refer Slide Time: 1:20)


Model: Store emails on server, index, ready to manipulate

View: Display list of emails; Read individual emails

Controller: Sort emails; delete; archive

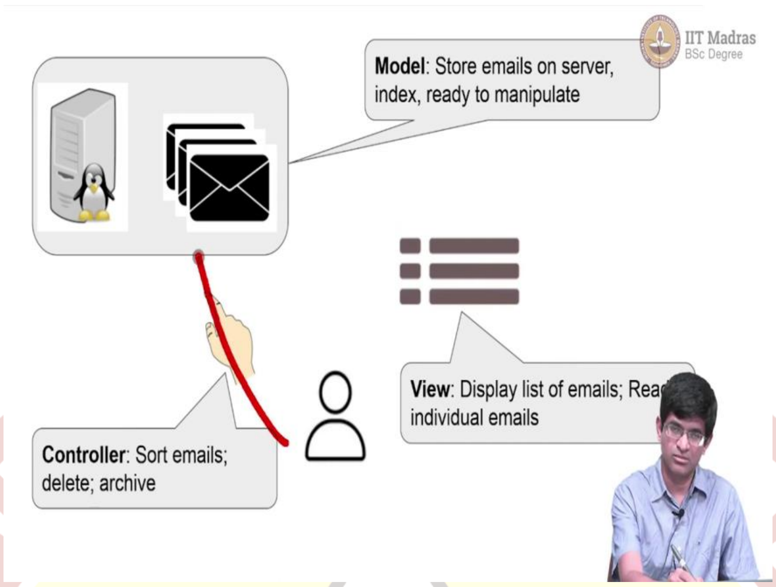

IIT Madras
BSc Degree
Model: Store emails on server, index, ready to manipulate
View: Display list of emails; Rea
individual emails
Controller: Sort emails; delete; archive

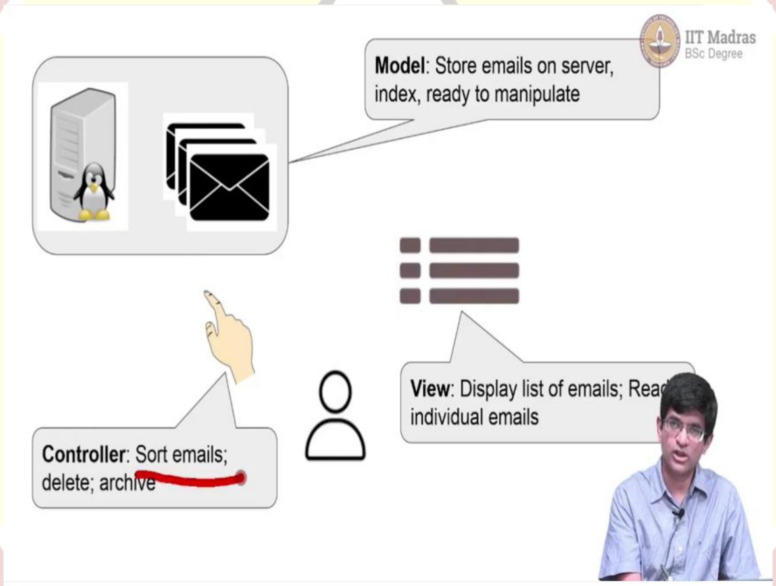
Let us quickly recap, what this is. The overall structure of a web app or in fact, any user interface based graphical user interface-based application can be thought of as having 3 components. The first is the model by which we basically mean, the data itself, which would typically be stored somewhere on a server, and has some kind of metadata or information associated with it.

If it is emails, it might have, for example, headers, such as the source, the destination, the time of sending the type of content, and maybe even some more details on how it was routed from the source to the destination. So, that is part of the model. In other words, all of that corresponds to data that needs to be stored on the server in order to understand what exactly it is that we are trying to manipulate over here.

The next part, the view, is where we would do for example, something like displaying a list of emails, reading individually, emails, maybe composing a new email, each of those would correspond to a different view, that is where the user interacts with the application. And the third part is the controller, which is the part that actually takes messages between the user interface that is the view and the data, which is the model.

(Refer Slide Time: 2:52)


IIT Madras
BSc Degree
Model: Store emails on server, index, ready to manipulate
View: Display list of emails; Read individual emails
Controller: Sort emails; delete; archive

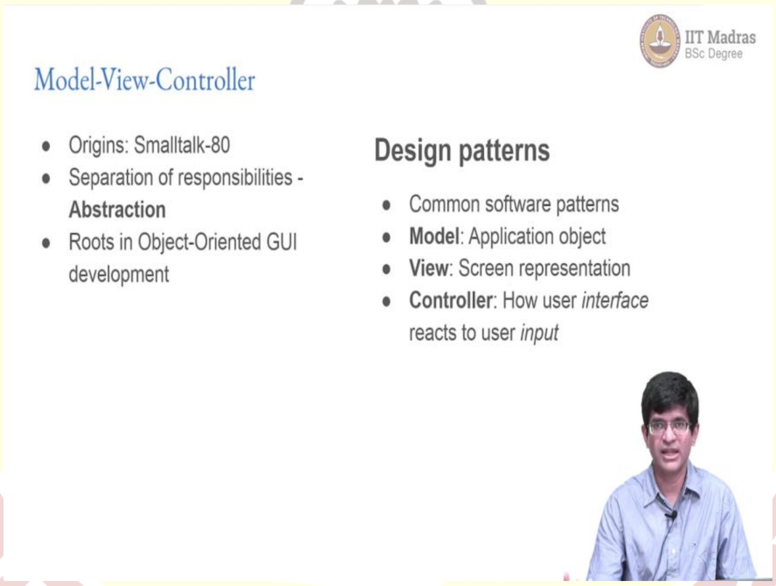

IIT Madras
BSc Degree
Model: Store emails on server, index, ready to manipulate
View: Display list of emails; Read individual emails
Controller: Sort emails; delete; archive

IIT Madras
BSc Degree
Model: Store emails on server, index, ready to manipulate
View: Display list of emails; Read individual emails
Controller: Sort emails; delete; archive

So, the controller is primarily responsible for taking information back and forth between the view and the model. For example, let us say that we wanted to sort the emails, a message would go from the view, where we will probably click on a button. But the message that goes to the server is that we want a different way of looking at the data, and perhaps do some filtering on the on the messages that are being sent back to the view.

Alternatively, the controller could actually manipulate the data. I might, for example, compose a new email, delete an old email, archive it set a certain flag, let us say make it important, or label it based on the person who sent it or according to the project under which it belongs. All of those are in some ways, manipulating the underlying model that is being stored in the data server. So, as I said earlier, in this lecture, we are going to concentrate on the view.

(Refer Slide Time: 3:51)



Model-View-Controller

- Origins: Smalltalk-80
- Separation of responsibilities - **Abstraction**
- Roots in Object-Oriented GUI development

Design patterns

- Common software patterns
- **Model**: Application object
- **View**: Screen representation
- **Controller**: How user *interface* reacts to user *input*

So, the core idea of model view controller has its origins in the late 70s, early 80s. In particular, there was a language called Smalltalk that came out of Xerox, the Palo Alto Research Corporation, Research Centre. So, Xerox PARC has a very important place in the history of modern computing, a lot of very interesting developments, both in terms of hardware and software came out of that lab.

Nowadays we know Xerox, mainly as you know, it is become synonymous with taking photocopies we take we say, take a Xerox whereas what we actually mean is take a photocopy of a piece of paper. So, Xerox PARC was a place where many developments and computing happened one of which was Smalltalk 80. And Smalltalk essentially was in some ways, a very interesting kind of object-oriented language.

And one of the things that they did as part of the development of Smalltalk was the separation of responsibilities, the abstraction, which allowed this whole object-oriented nature, you had certain objects which could maybe have a window as an object, you could also have an object which represented some kind of data, the model that you are trying to manipulate. And each of those was then essentially manipulated by sending messages from one part to another.

So, this whole idea of the model view controller has its origins over here, but since then has come to be implemented extensively in various forms of general user interfaces. And, in fact, one way of looking at it is that the MVC is not so much a design pattern by itself or a design paradigm by itself, it is more a combination of a few different software patterns.

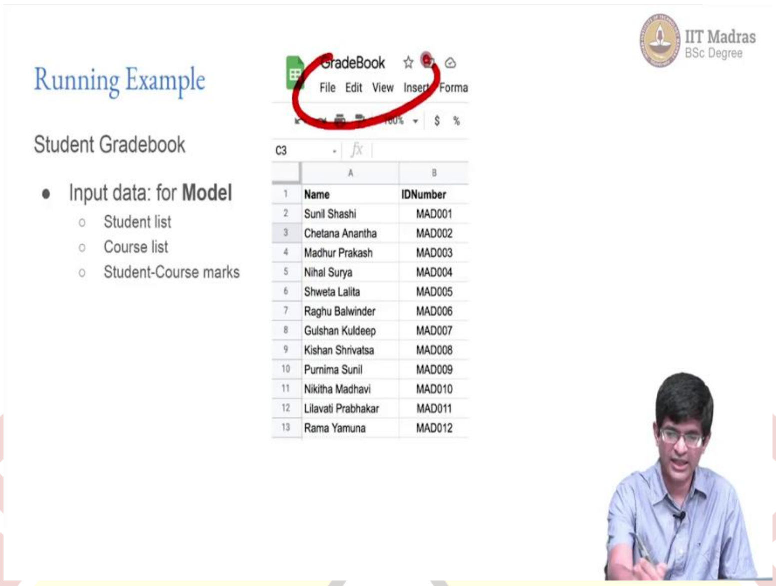
So, design patterns, as I had briefly mentioned once earlier, are essentially common software patterns, they represent in some sense experience. Several software designers, over the years have found that certain ways of implementing logic are more efficient than others, or are very effective at getting certain properties implemented. And those learnings over the years are abstracted out into various kinds of design patterns.

So, according to this, essentially, the model represents the application object, whatever it is that you are trying to manipulate. The view is the screen representation. And each of those themselves the model as well as the view can be thought of as an object, in some ways. The view is an object because it has certain properties of its own, where is it being displayed? What size is the screen on which it is being displayed? What kind of buttons are present on it? How are they located? The layout? All of those are properties of the view.

Similarly, the model has its own properties. Who sent the email to whom was it sent? When was it sent through which router was it passed? All of those are properties of the email. So, this separation of the design patterns allowed the construction of this model view controller. And ultimately, the controller part is the messages that go between these two, the view and the model. It can be thought of as essentially, how does the user interface react to user input.

And this is something that you need to keep in mind, even when you are developing a web app, there are two parts where the user is interacting with your application. One is they are consuming what you show them, they are actually seeing what you put on the screen. The other is they are actually giving you inputs, they are pressing buttons, they are clicking on links, or they might be entering text into text boxes, all of those are the user input, which ultimately determines what comes back and is presented for user interaction.

(Refer Slide Time: 7:48)



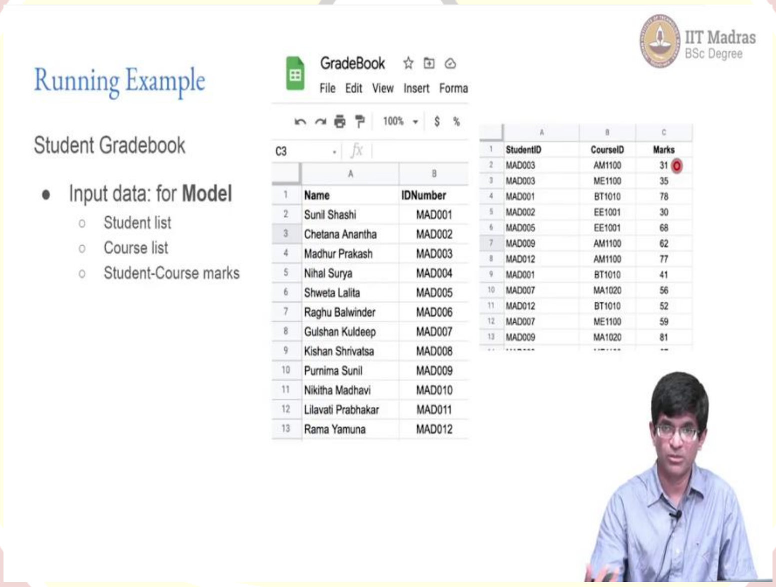
Running Example

Student Gradebook

- Input data: for **Model**
  - Student list
  - Course list
  - Student-Course marks

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



Running Example

Student Gradebook

- Input data: for **Model**
  - Student list
  - Course list
  - Student-Course marks

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	C
1	**StudentID**	**CourseID**	**Marks**
2	MAD003	AM1100	31
3	MAD003	ME1100	35
4	MAD001	BT1010	78
5	MAD002	EE1001	30
6	MAD005	EE1001	68
7	MAD009	AM1100	62
8	MAD012	AM1100	77
9	MAD001	BT1010	41
10	MAD007	MA1020	56
11	MAD012	BT1010	52
12	MAD007	ME1100	59
13	MAD009	MA1020	81

So, through this series of lectures, I am going to be following one running example, which allows us to sort of bring out most of these in more concrete forms. And the example that I am going to use is a student grade book. And what I mean by that is, you have a list of students, and a list of courses, so there are some, I am going to basically generate names and data at random or here.

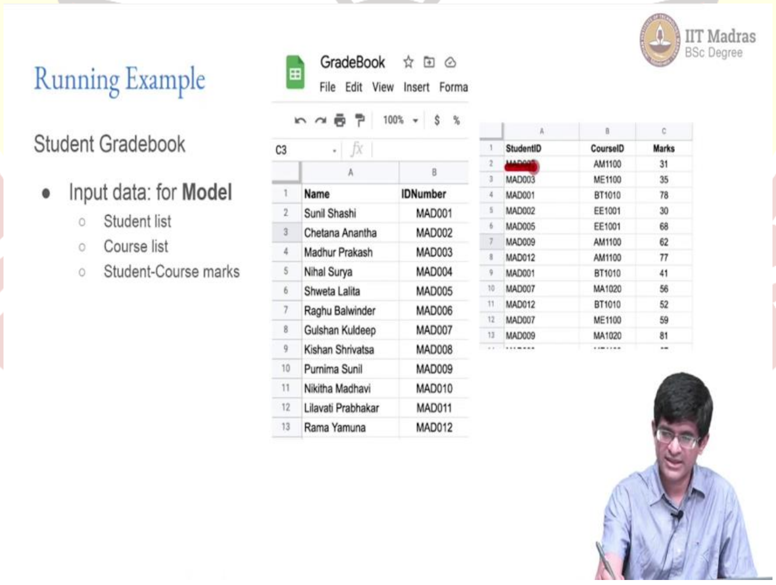
And it does not really correspond to any real person or real set of courses. Literally, what I do is go into a spreadsheet, use the random number generator and create a bunch of names, courses, and marks. But this is representative of an actual application, because we might have something like this, where you have a list of students a list of courses, and there are some other entity out there that allows you to represent the relationship between the two. All of this is part of how the model is implemented. And we will get to that in a future lecture.

Part of what we are going to do in terms of storing the model at least right now the way to look at it is, just a spreadsheet. As you can see, this is what it would look like I have something created in Google Sheets. It has a list of names out here. And each of these names has some kind of an ID number associated with them.

Now, are the ID numbers necessary, are the names necessary? What part of this is absolutely necessary for the system to work? That is part of the design of the system. It is possible to work just with names, you do not really need ID numbers. Of course, the problem is you might have two people with the same name.

So, in any school or college, what you would normally find is that people are given distinct roll numbers or ID numbers to identify them. So, all that this application is going to use is that you have a name and an ID number associated with the person. And the third part, the student course marks, similarly there would be something for courses they would have names and the courses would also have some kind of an ID number that makes it easy to identify the course.

(Refer Slide Time: 10:01)



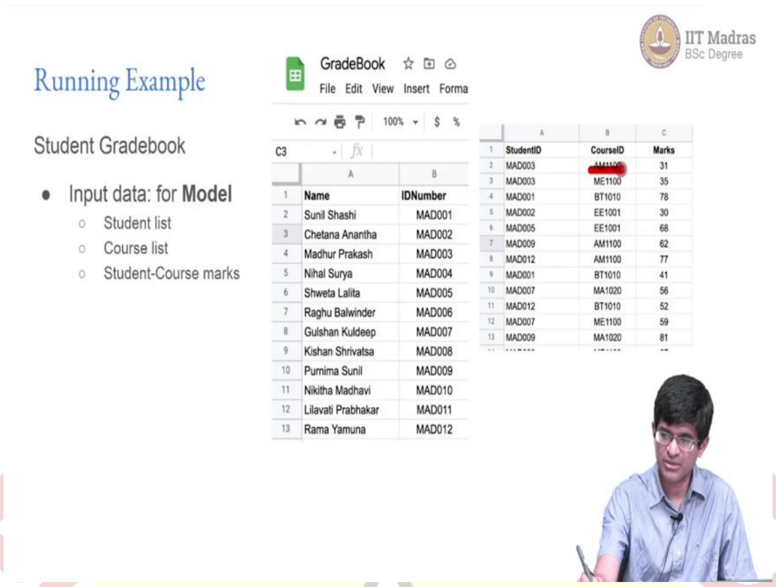
Running Example

Student Gradebook

- Input data: for **Model**
  - Student list
  - Course list
  - Student-Course marks

	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	C
1	**StudentID**	**CourseID**	**Marks**
2	[illegible]	AM1100	31
3	MAD003	ME1100	35
4	MAD001	BT1010	78
5	MAD002	EE1001	30
6	MAD005	EE1001	68
7	MAD009	AM1100	62
8	MAD012	AM1100	77
9	MAD001	BT1010	41
10	MAD007	MA1020	56
11	MAD012	BT1010	52
12	MAD007	ME1100	59
13	MAD009	MA1020	81



Running Example

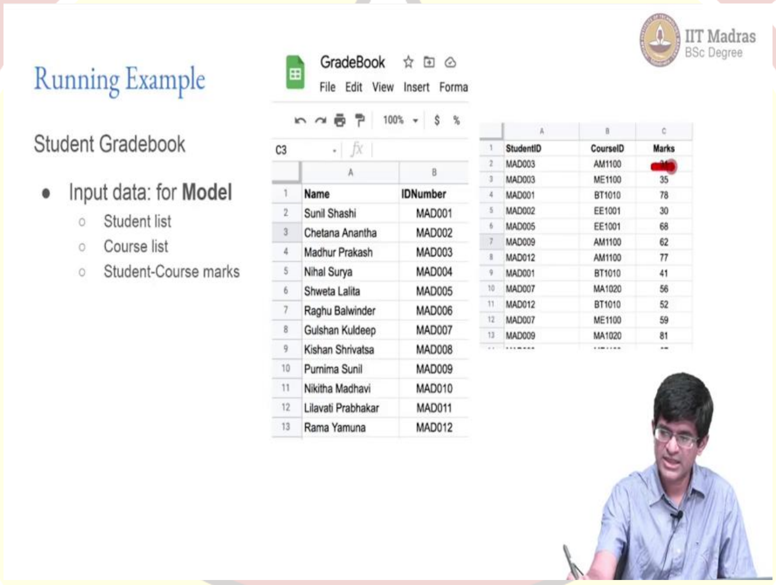
Student Gradebook

- Input data: for **Model**
  - Student list
  - Course list
  - Student-Course marks

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	C
1	StudentID	CourseID	Marks
2	MAD003	AM1100	31
3	MAD003	ME1100	35
4	MAD001	BT1010	78
5	MAD002	EE1001	30
6	MAD005	EE1001	68
7	MAD009	AM1100	62
8	MAD012	AM1100	77
9	MAD001	BT1010	41
10	MAD007	MA1020	56
11	MAD012	BT1010	52
12	MAD007	ME1100	59
13	MAD009	MA1020	81



Running Example

Student Gradebook

- Input data: for **Model**
  - Student list
  - Course list
  - Student-Course marks

GradeBook

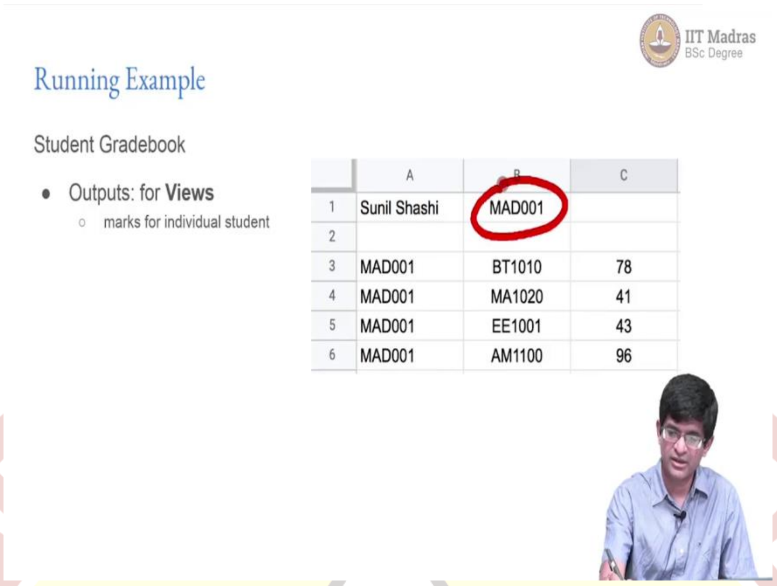
	A	B
1	Name	IDNumber
2	Sunil Shashi	MAD001
3	Chetana Anantha	MAD002
4	Madhur Prakash	MAD003
5	Nihal Surya	MAD004
6	Shweta Lalita	MAD005
7	Raghu Balwinder	MAD006
8	Gulshan Kuldeep	MAD007
9	Kishan Shrivatsa	MAD008
10	Purnima Sunil	MAD009
11	Nikitha Madhavi	MAD010
12	Lilavati Prabhakar	MAD011
13	Rama Yamuna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

	A	B	C
1	StudentID	CourseID	Marks
2	MAD003	AM1100	31
3	MAD003	ME1100	35
4	MAD001	BT1010	78
5	MAD002	EE1001	30
6	MAD005	EE1001	68
7	MAD009	AM1100	62
8	MAD012	AM1100	77
9	MAD001	BT1010	41
10	MAD007	MA1020	56
11	MAD012	BT1010	52
12	MAD007	ME1100	59
13	MAD009	MA1020	81

And you would have some kind of a link, there is an ID number for the student and then ID number for the course. And something which indicates the marks in that course. So, by that I can basically say that this particular student corresponding to ID numbers 003. In this course, AM1100 got 31 marks. So, in that way, of course, the data that I have shown over here, it is possible that there might be duplicates, like I said, I just generated some random values.

But in general, you would need to also do some validation on the data, make sure that you do not have, for example, different marks for the same student for the same course, you cannot have more than one entry. All that comes later when we look at the model.

(Refer Slide Time: 10:41)

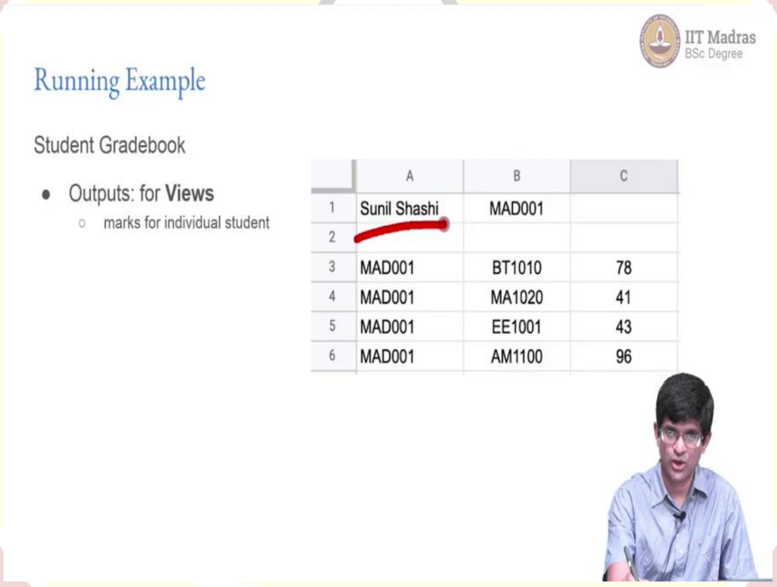


Running Example

Student Gradebook

- Outputs: for **Views**
    - marks for individual student

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96

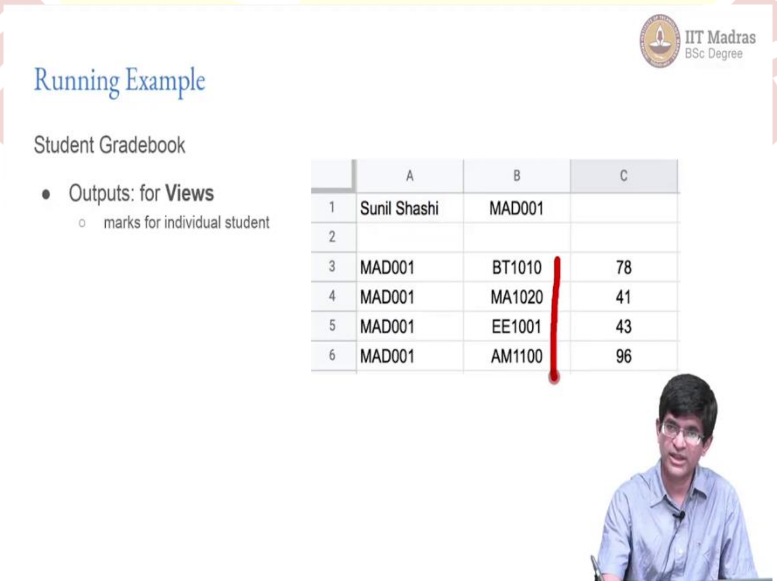


Running Example

Student Gradebook

- Outputs: for **Views**
    - marks for individual student

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96

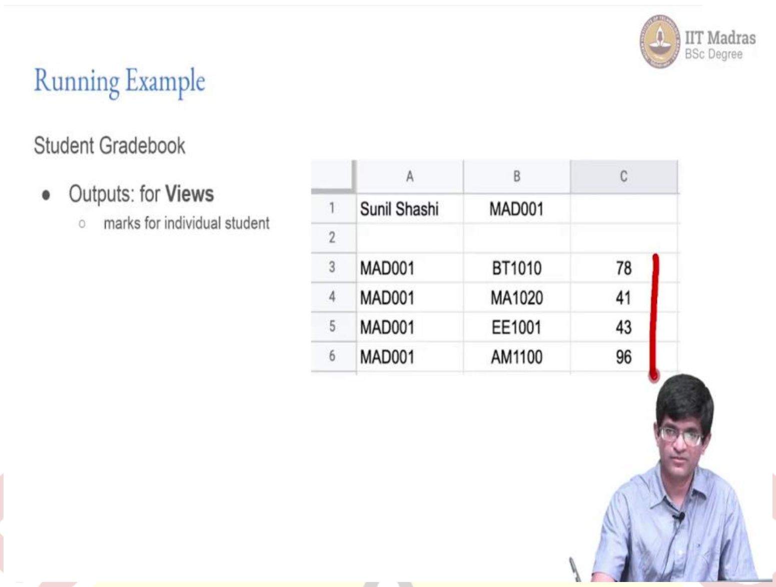


Running Example

Student Gradebook

- Outputs: for **Views**
    - marks for individual student

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96



Running Example

Student Gradebook

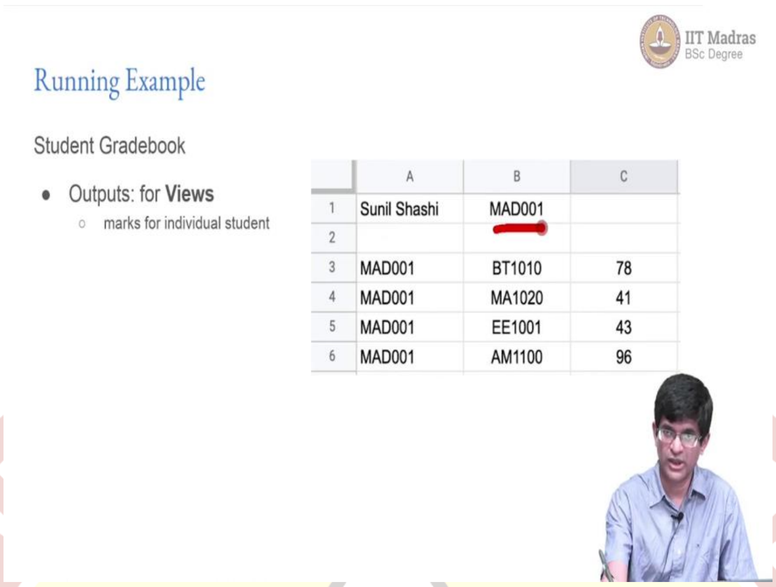
- Outputs: for **Views**
  - marks for individual student

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96

For now, what we are interested in is the view. And possible views of this could be something like this, let us say that I want all the marks for one individual student, I might, for example, have another spreadsheet or another location, where I can enter the ID number of the student. Ideally, the system should then pull out the name from the student database and show it to me. And it should also say that these are the courses the student is registered for. And these for the marks in those courses.

Now, in this particular case, I have shown this I have done this by hand, I basically pulled out the name I pulled out corresponding to this number. And I just copied and pasted the entries corresponding to the courses that that student had taken. Is there a way to automate this, of course, even within the constraints of a spreadsheet, it should have been possible for me to create something using some lookups and other kinds of techniques, which would allow me to create such a table like this. But as far as we are concerned right now, the important point is this is an example of a view.

(Refer Slide Time: 11:48)

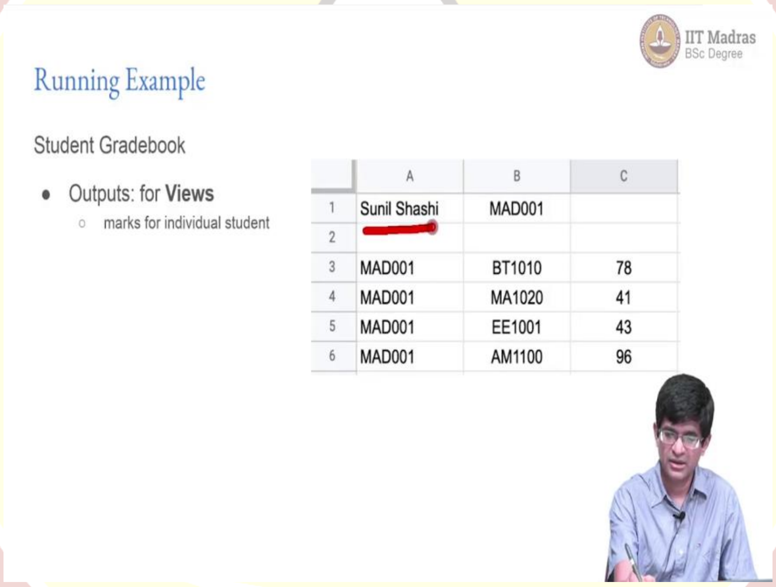


Running Example

Student Gradebook

- Outputs: for **Views**
  - marks for individual student

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96

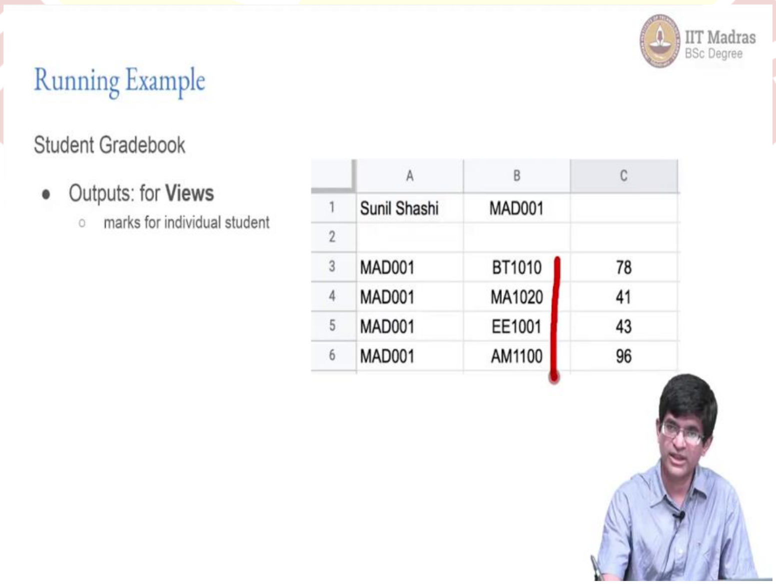


Running Example

Student Gradebook

- Outputs: for **Views**
  - marks for individual student

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96

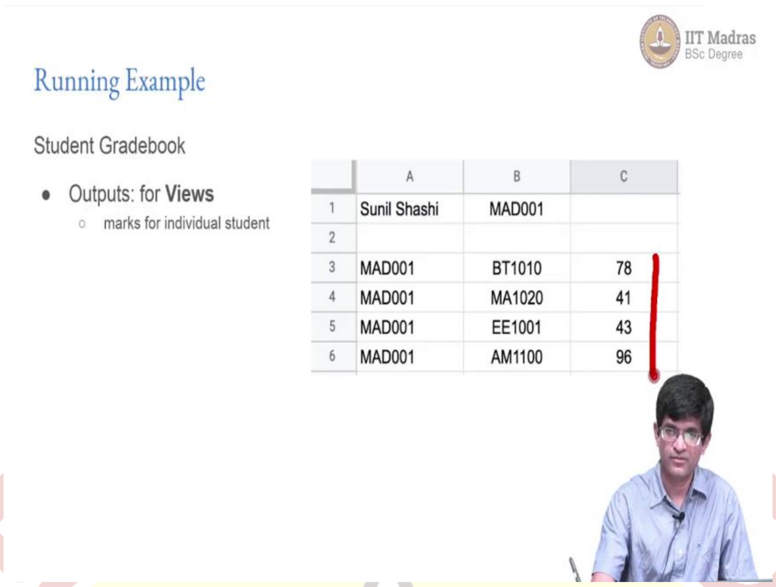


Running Example

Student Gradebook

- Outputs: for **Views**
  - marks for individual student

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96



Running Example

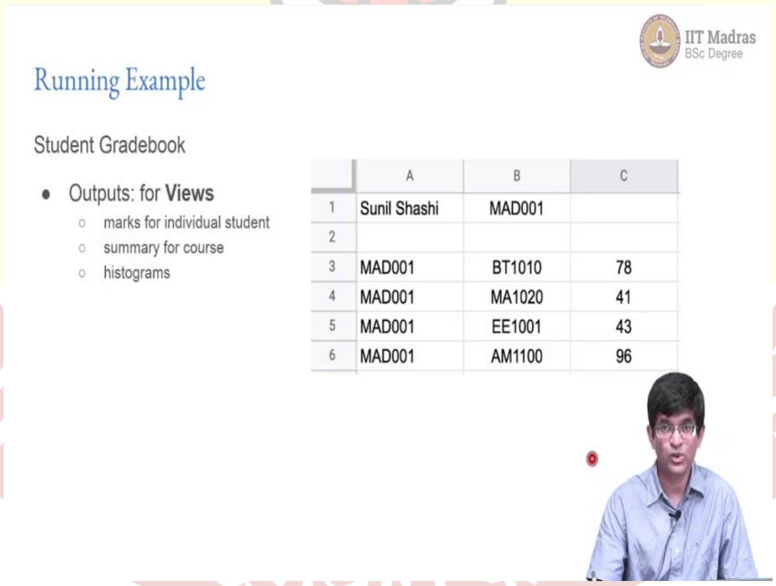
Student Gradebook

- Outputs: for **Views**
  - marks for individual student

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96

So, in this case, I have a view where I provided as input, a user ID number, and I have as output the name of the person, the courses for which they are registered and the marks that they got in each of those courses. In this particular case, the view was constructed by hand. And obviously, what we are interested in is can this be done automatically?

(Refer Slide Time: 12:10)



Running Example

Student Gradebook

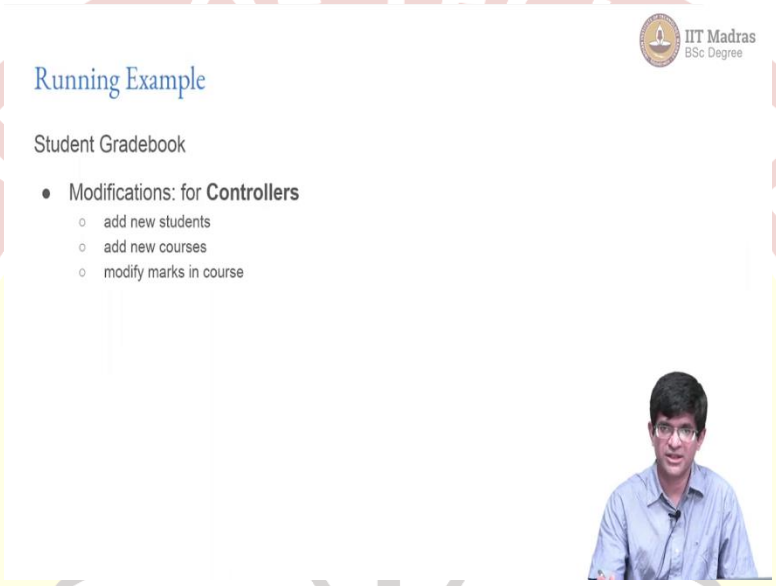
- Outputs: for **Views**
  - marks for individual student
  - summary for course
  - histograms

	A	B	C
1	Sunil Shashi	MAD001	
2			
3	MAD001	BT1010	78
4	MAD001	MA1020	41
5	MAD001	EE1001	43
6	MAD001	AM1100	96

Now, there are many other kinds of views you could think of as well. So, for example, if I gave a course number as input, I might like to see how many students were registered for it? What was the distribution of marks? Who got the highest score? Who got the lowest score? And does the distribution sort of follow a bell curve? At what point should I set the cut-off for a high grade or a top grade or the next possibly grade and so on? I might want to look at histograms, I might want to look at plots of the data.

So, in other words, you can already start imagining that the view that a person is looking for, could be a table, it could be a plot, it could be plain text, it might even be in some other format, which is you know, some kind of structured data, like Java Script Object Notation or JSON. Why JSON because that way I could take that data and feed it into something else, some other programme that I write in order to represent it in some other way that I find convenient. So, each of these are different options as far as views are concerned.

(Refer Slide Time: 13:13)



Running Example

Student Gradebook

- Modifications: for **Controllers**
  - add new students
  - add new courses
  - modify marks in course

And finally, as far as controller, the last part of it the controllers, we will get to that much later. What we will be looking at is, you know, for example, how do you add new students, add new courses, modify your course, modify a student's information, enter the marks for a student in our course. All of those are examples of controller functionality that need to be implemented for this entire grade book application to be useful.