INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •

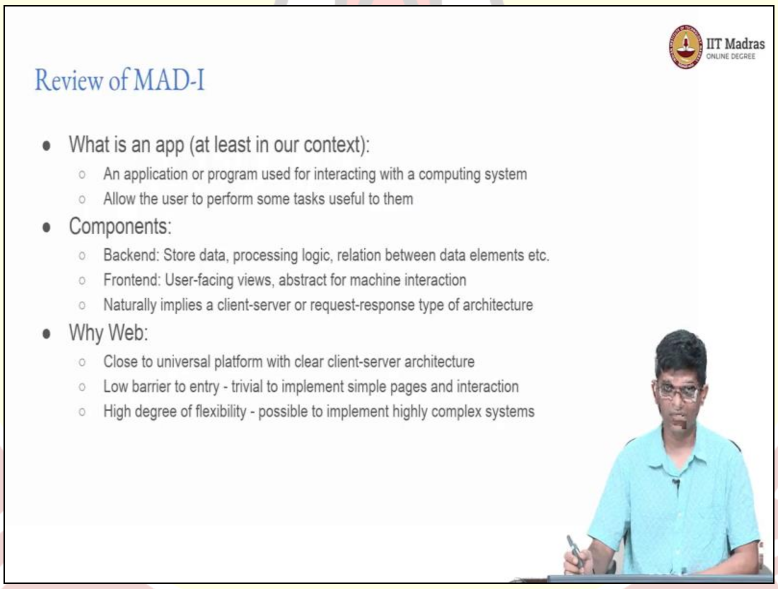
IIT Madras
ONLINE DEGREE

**Modern Application Development II**
**Online Degree Programme**
**B. Sc in Programming and Data Science**
**Diploma Level**
**Prof. Nitin Chandrachoodan**
**Department of Electrical Engineering**
**Indian Institute of Technology- Madras**

## Review of Modern Application Development - I and what next?

Hello everyone welcome to modern application development part 2. So, as you might expect this is a follow-up course to the course on modern application development and what I am going to do is to first start off with a quick overview of what are the topics that were covered in the first course the application development one course. And sort of set the expectations in terms of what we might be seeing as part of the advanced course.

**(Refer Slide Time: 00:38)**



Review of MAD-I

- What is an app (at least in our context):
  - An application or program used for interacting with a computing system
  - Allow the user to perform some tasks useful to them
- Components:
  - Backend: Store data, processing logic, relation between data elements etc.
  - Frontend: User-facing views, abstract for machine interaction
  - Naturally implies a client-server or request-response type of architecture
- Why Web:
  - Close to universal platform with clear client-server architecture
  - Low barrier to entry - trivial to implement simple pages and interaction
  - High degree of flexibility - possible to implement highly complex systems

So, a quick review of the modern application development part one. We started off by looking at the concept of an app what is an app or at least in our context in the context of this course. And what we defined it was essentially it is an application or a program that we use for interacting with a computing system that is a very generic and broad definition.

It allows the user to perform some tasks that are useful to them and that ultimately is what we are interested in. We the user the end user should be able to use the application in order to do something that was relevant to them. And there are obviously multiple ways by which this can be done and as part of what we saw in the course of course you know there was this thing that came up as you know what is an application versus what is an app?

We are not really drawing very strict distinctions over there. What we saw at a high level was that an application can be broadly divided into a few components in terms of the structure how you are going to look at it. And from there what we said was primarily there is something called the back end where the application needs to store data and the processing logic whatever inputs come from the user.

How do we process the data that is present in the system based on those inputs? What are the relationships between the relationships between various data elements present that are being stored as part of the app. All of those are part of the back end. And then we come to the front end which is the user facing view the part that the end users actually interact with.

It might not only be a human user it might also be a machine user meaning that there might be machine consumers of the information produced by the app and there we saw certain ways by which we can also perform some abstractions that make the machine to machine interaction also cleaner than what it would be normally otherwise. Now the moment that we start looking at an app in this way it naturally implies a client server kind of view of the system.

There are other ways of looking at applications but for our purposes and especially given the context in which we are looking at it a client server or a request response kind of behavior is the cleanest abstraction that we can use in order to understand how applications are built. And of course we chose the web as the platform on which we were building these applications.

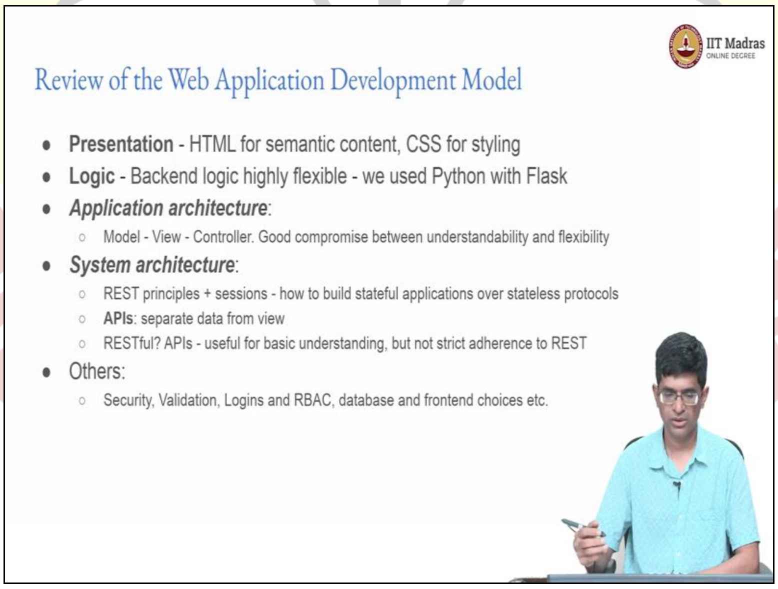
Why the web? Well today it is close to a universal platform in terms of its reach. It is available on desktops on mobile platforms on tablets even in kiosks. So, it is probably the easiest platform that we can think of if we want to deploy an app that is to reach as many people as possible not only that the web by its very nature by design has a very clear client server architecture.

The entire sort of structure of the web is built around that request response kind of behaviour. It has a very low barrier to entry meaning that it is almost trivial to implement a page that just has some simple html pages some kind of styling and maybe minimal interaction some kind of a form that you need to fill out and click on a submit button. So, getting a very basic application going with the web based platform is simpler probably than any other kind of application that you might think of that actually gets deployed on a operating system.

There you at least need a compiler you need an SDK you need to put the entire system together whereas over here creating an html page and the corresponding styling and the forms is all you need is a text editor. So, the barrier to entry in terms of the web is very low. On the other hand it also gives you a very high degree of flexibility meaning that you can if you want to implement really complex applications using the same platform.

And you can actually build systems where multiple different parts are interacting with each other.

**(Refer Slide Time: 04:50)**



Review of the Web Application Development Model

- **Presentation** - HTML for semantic content, CSS for styling
- **Logic** - Backend logic highly flexible - we used Python with Flask
- ***Application architecture***:
  - Model - View - Controller. Good compromise between understandability and flexibility
- ***System architecture***:
  - REST principles + sessions - how to build stateful applications over stateless protocols
  - **APIs**: separate data from view
  - RESTful? APIs - useful for basic understanding, but not strict adherence to REST
- Others:
  - Security, Validation, Logins and RBAC, database and frontend choices etc.

Now the application development model that we used as part of this course was we first started off by looking at the presentation aspects. Where we saw that html is used for what's called the semantic content meaning that the tags in html like the h1 tag for a heading or the a tag for a link or a p tag to indicate a paragraph or a div tag to indicate a section of text all of those have some kind of semantic meaning.

And if you use them properly you can actually convey meaning even beyond what is being displayed on the screen and that meaning could be consumed or interpreted in different ways. So, for example by looking at all the h1 tags it is easy to pull out a table of contents whereas if you had just gone around making some text bold and large font it is not entirely clear to a computer whether you actually meant it to be a heading or just you wanted to emphasize it.

So, html allows us to do semantic marker and of course CSS cascading style sheets provide immense flexibility in terms of how you can style it you can make it look different in each case. And in terms of the backend logic that is the part that actually implements the business logic. So, to say, where you get the inputs from the user and you have to do something with those inputs and generate outputs.

We use python with the flask framework. Python being an immensely flexible language and relatively easy to learn and flask being a very simple framework which pretty much does not get in our way and allows us to focus on the basic concepts that we are trying to implement. Of course we also went through and said yeah there are other frameworks there are other languages that you can use there are various different preferences that people might have.

So, this the reason why we chose python and flask was more for simplicity in terms of how we explain concepts. And you are perfectly free to then go on and develop similar kinds of applications using your preferred platform. So, after the basic structure that is you know you have the front end which is html and CSS and the back end logic implemented with python.

We also looked at two other aspects one is called the application architecture and in terms of the application architecture we basically used what is called the model view controller. Once again this is a good compromise between the ease of understanding and the flexibility that it provides. And as discussed App dev-1 we do have this you know the implication of using this model view controller is that is one particular way of looking at application architectures.

And there are a number of different variants that exist today. Once again the reason why we chose MVC is because it is fairly solid it has like good groundings in terms of the understanding of how it was designed and implemented. It applies very well to generic graphical user interfaces. There may be specific instances of some kind of you know application platform where maybe this is not the best possible way of looking at it but it is still a very good way of separating out the different concerns that are made.

The next thing that we looked at was so called system architecture and over here we brought in this concept of risk. The representational state transfer and idea of sessions how to build stateful applications using stateless protocols. And one very important concept over there over there was this idea of an api or an application programming interface. The fundamental idea of an api was that we are now separating out the data that is transferred between the server and the client from how it is actually displayed.

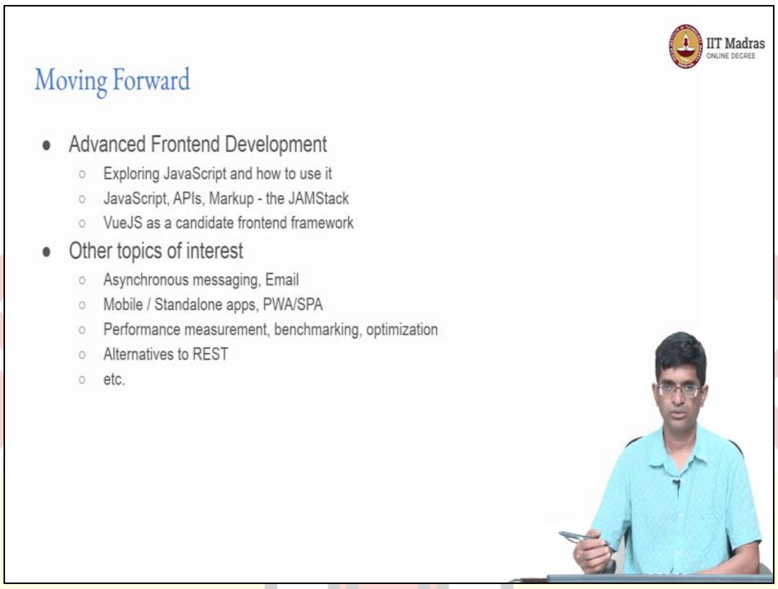
And we will see later in App dev-2 that we will be making extensive use of api's and primarily sort of saying how do we get a front end that can be integrated with various different kinds of api's in order to give you the look and feel of the application that you want. There is also this concept of rest and the restful nature and restful api's the reason I put a question mark over there is because there is a lot of debate about whether some of the concepts that are even described as restful.

Actually apply directly in the context of the original definition of rest. Where there are like many features that are fairly very clearly defined which may not always apply in what we think of as a restful api. So, of course we use it primarily in the context of things like correct the create read update delete for different models, rest goes beyond that. So, we looked at it to some extent but we did not want to get into too much detail on how exactly this is to be implemented.

That is not the purpose of this course what we are looking at is what do we need to understand in order to get a basic system level architecture. So, we are not strictly adhering to rest but on the other hand we are using these concepts of api's and so on. App dev one also covered several other topics including security, validation of data how to implement logins and role-based access control database and front-end choices all of those were sort of touched upon in different details.

And in different levels of detail and finally the whole thing was sort of you know put together in the form of a project that was interesting.

**(Refer Slide Time: 10:27)**



## Moving Forward

- Advanced Frontend Development
  - Exploring JavaScript and how to use it
  - JavaScript, APIs, Markup - the JAMStack
  - VueJS as a candidate frontend framework
- Other topics of interest
  - Asynchronous messaging, Email
  - Mobile / Standalone apps, PWA/SPA
  - Performance measurement, benchmarking, optimization
  - Alternatives to REST
  - etc.

So, now where do we go from there right. So, moving forward what we are going to be looking at primarily in terms of app dev 2 is we are going to focus a lot more on advanced front end development. And what I mean by that is the third element out of the html css and the third javascript which we only touched upon very briefly in app dev one is going to be one of the primary focuses at least for a major part of app dev 2.

Now why is this why exactly is javascript such a big deal and why do we need to spend such a lot of time on it. Potentially there might be ways you know I mean people might say that yeah you know javascript is not really the best language but on the other hand just given the nature of its interaction with the web it makes it an immensely powerful way of building really good applications that are directly deployable on a web-based platform.

We will go through many of the features that make this possible as we understand a little bit more about javascript. And a lot of what we will be looking at is. So, you as you try and build up you know better front ends using the javascript how do we then integrate that with api's which we have already looked at to some extent and use that in order to generate markup that finally gets displayed on a screen.

And this combination javascript api’s and markup is sometimes referred to as the JAMStack I will not be really using this term a lot but it may be something that you come across. And one of the things that we will do of course we will first start by going over javascript basics but then we will look at one potential candidate for a front-end framework. And in this case what we are going to be doing is using this platform called Vue, VueJS, Vue javascript.

Now once again just based on popularity you might argue that perhaps react or angular have more users but that is not the primary focus of what we are looking at what we are trying to find is a balance between usefulness meaning that it is popular and known to work well versus ease of explaining certain concepts and also the ease of getting started with it. Vue provides us with some advantages on that front which is why we are going to go with that approach.

Now apart from this the front end we will also be looking at several other aspects that come into play in developing an application. Some of it including things like asynchronous messaging how do you sort of schedule emails for delivery how do you communicate over emails because there are issues that come up over there that need to be addressed properly.

And in particular it comes to the nature of the how exactly are you going to respond to a request. If I am going to do heavy duty computations and sending emails and so on in a way that I actually try and you know as soon as a user clicks on the button it actually has to send the email and then come back. You will see that you know that can potentially lead to big delays and poor user experience.

So, we need some kind of asynchronous concepts that come in over there and that is going to be sort of a running thread throughout this course. We will also look briefly at how we can take this you know this whole JAMStack kind of approach and use it to as one approach to build some kind of mobile as well as stand-alone applications and there are also some concepts like progressive web apps single page applications all of which we will at least touch upon in order to understand the concepts.

Some part definitely needs to be spent on performance measurement, benchmarking and optimization. How do you find out whether your site is good or bad in terms of the speed of response that is one thing that we will be spending some time on and possibly some alternatives to rest one well known and you know upcoming alternative is what is called Graphql.

So, we will look at those as well as potentially other topics as the course progresses. But a large part of the beginning is going to be devoted to javascript, exploring javascript, how to use it and advanced uses of javascript.