Design and Implementation of Human – Computer Interfaces

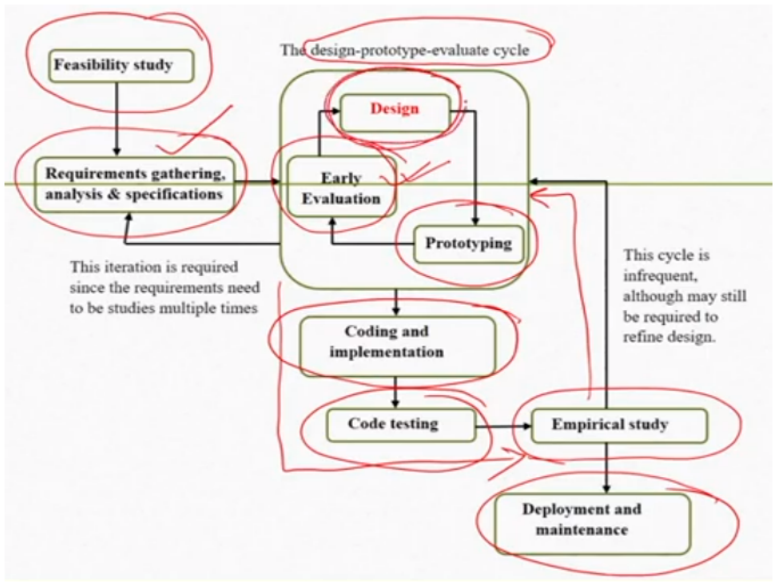
Prof. Dr. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module No # 03
Lecture No # 13
Schneiderman's Golden Rules

Hello and welcome to the NPTEL MOOCs course on design and implementation of human computer interfaces. We are going to start lecture number 12 on design guidelines. Before we start let us quickly recap what we have learned so far so as the name of the lecture indicates, so we are going to talk about guidelines for design. Design of what and where in the overall interactive system development life cycle the design fits.

We have so far learned about the life cycle and the different stages of the life cycle. So, let us just quickly recollect what are the stages? And where we are currently now?

(Refer Slide Time: 01:34)


Feasibility study
The design-prototype-evaluate cycle
Design
Requirements gathering, analysis & specifications
Early Evaluation
Prototyping
This iteration is required since the requirements need to be studies multiple times
This cycle is infrequent, although may still be required to refine design.
Coding and implementation
Code testing
Empirical study
Deployment and maintenance

So, if you may recollect, we have several stages in the interactive system development life cycle. We start with the requirement gathering analysis and specification stage. Feasibility study although it is shown as part of the life cycle actually is a stage where we decide whether to proceed or not? Or if we proceed then whether; any modification in the plan is required or not so. As such it is not directly involved in the development of the system.

So, we will treat it separately and in this course, we are not discussing in detail the feasibility study stage. We started our discussion with requirement gathering analysis and specification, which we considered the first stage in the interactive system development life cycle. After this, we enter the design prototype evaluation cycle. Now, this cycle contains 3stages the design stage, the prototyping stage, and the evaluation of the prototyping stage.

Now here, we mentioned that design involves 2 types of design the design of the interface and interaction is a key component in any interactive system development and also the design of the actual system or design of the code. So, when we are talking of the design prototype evaluation lifecycle, we are primarily referring to the design of the interface and interaction rather than the code design. For code design, we may not require this cycle.

So, in the interface and interaction design, we come up with a design, then prototype it. Because we are dealing with user center design approach so, we need to take into account the user inputs in as many stages as possible. So, to take into account user feedback, we create a prototype of our design, then get it evaluated. If any issues are found, then we refine our design, recreate the prototype, and get it evaluated again and this goes on in a cycle which we call the design prototype to evaluate cycle.

So, once we reach a stable design that means a design, where not many significant or new issues are found through prototyping and evaluation, then we stop the cycle there and enter the design of the coding stage. Now, in the design of the code, we plan for the overall software system. So, to speak and then we go to the next stage. The next stage is coding and implementation. That means once we have a design of the system ready, we go to implement it by writing programs that is the coding stage.

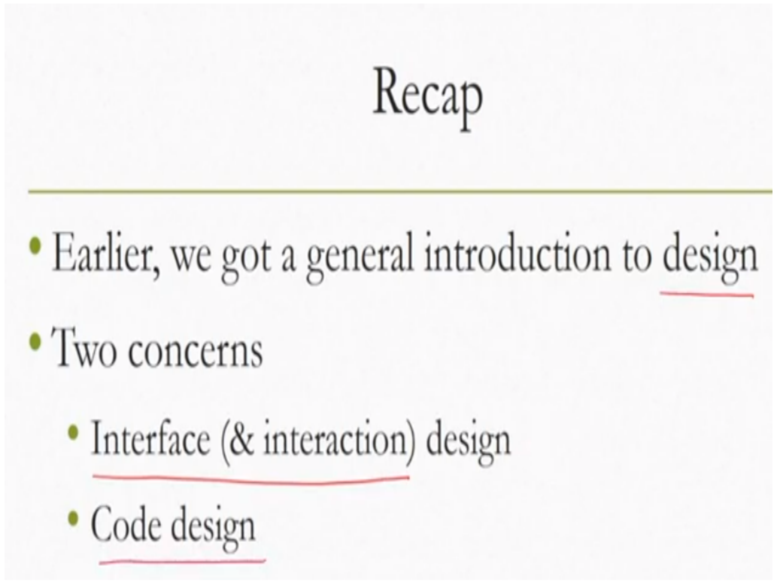
After we have implemented it, we need to test it. So, this is the code testing phase which comes after the implementation stage. Now, in code testing of course there are several levels of testing and several types of testing, which are related to the testing of the code. Note that in the evaluation phase we talked about in the design evaluates prototype cycle here, it is related to the evaluation of the design from the point of view of usability.

Now when we are talking of code testing, we are talking of evaluation of the code from the point of view of execution ability. In other words, whether; the code can be executed efficiently and as per the expectations. So, after code testing, we go for another round of usability testing now this time for the whole system rather than a prototype, which we call empirical study or empirical research.

Now here, what we do is test the usability of the system with respect to end users and if we find some problems at this stage which is possible, then we may like to go back to the design stage and traverse the other stages again. So, this can form another cycle although this should be a minimal maximum of one or twice, more than that will affect the overall turnaround time as well as the cost of the project.

So, once we are assured of the usability of the product as well as the execution ability of the product we go for deployment and subsequently maintenance stage of the life cycle .So, this is the summary of what we are talking about we have already discussed the requirement gathering stage and currently we are discussing the design of the interface stage.

(Refer Slide Time: 06:50)



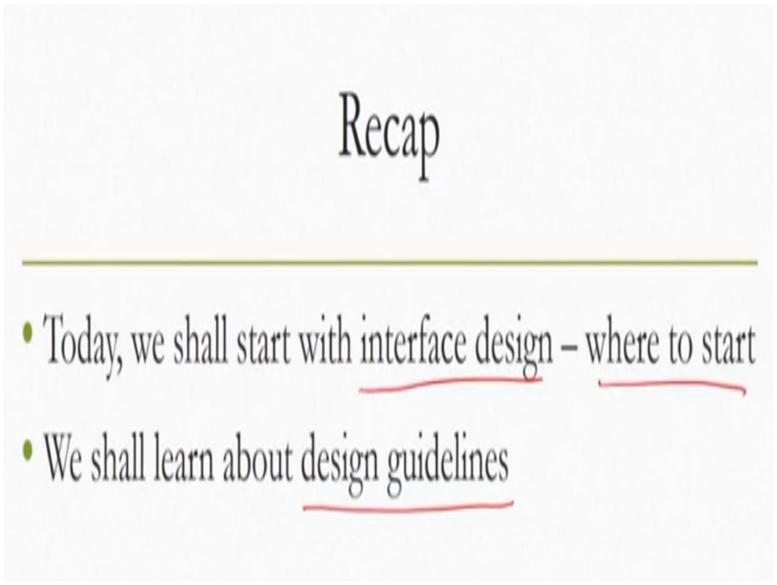
Recap

- Earlier, we got a general introduction to design
- Two concerns
  - Interface (& interaction) design
  - Code design

In earlier lectures, we got a general introduction to the problem of design. What are the issues and how do we address those issues in the design of the interface? Now, there again we talked of

2 design problems here, one is interface and interaction design where usability is a prime concern and code design or design of the system.

(Refer Slide Time: 07:25)

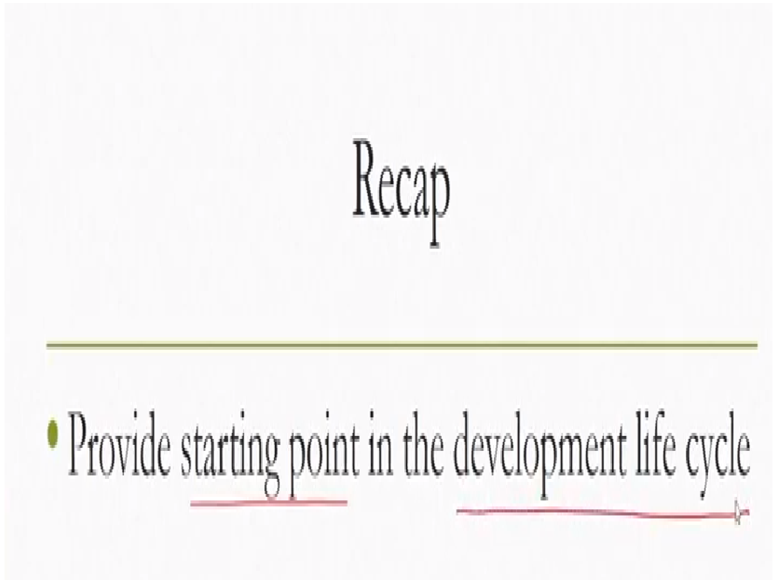


Recap

- Today, we shall start with interface design – where to start
- We shall learn about design guidelines

Now in this lecture, we are going to talk about the design of the interface where to start the process that is going to be our primary concern in this lecture. In this regard we are going to learn about design guidelines or guidelines for the design of interfaces and interactions so what are the things that we should follow while going for a design?

(Refer Slide Time: 07:55)



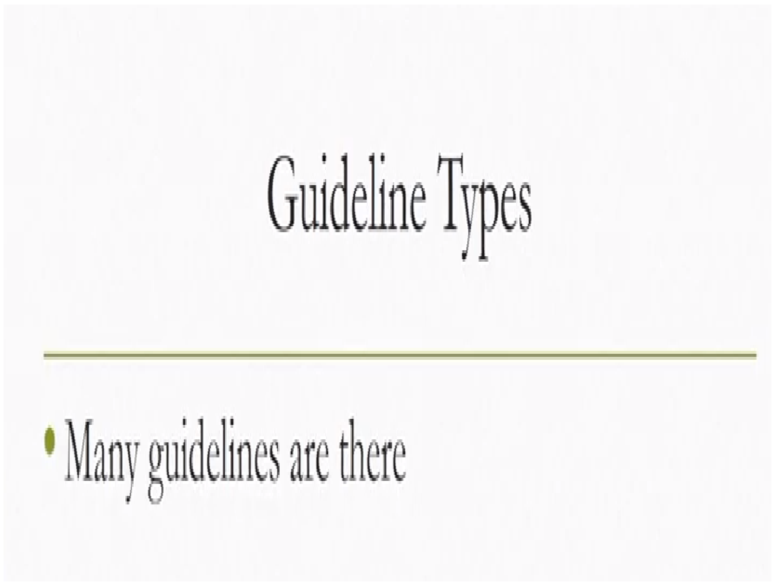
Recap

- Provide starting point in the development life cycle

When we are talking of design, if you may recollect 2 issues are of primary concept. One is where to start and the second one is how to specify? The same applies to interface design where we should start our design process? Now, if you are an experienced designer and also you have sufficient experience in the design of similar systems, then you can rely on your experience which again is influenced by your intuition and knowledge level, and skills to start the process.

Even if that is the case or if you do not possess such experience, then also we can start by taking recourse to guidelines. So, guidelines provide a starting point in the design phase of the development life cycle earlier we talked about several guidelines.

(Refer To Slide Time: 09:08)

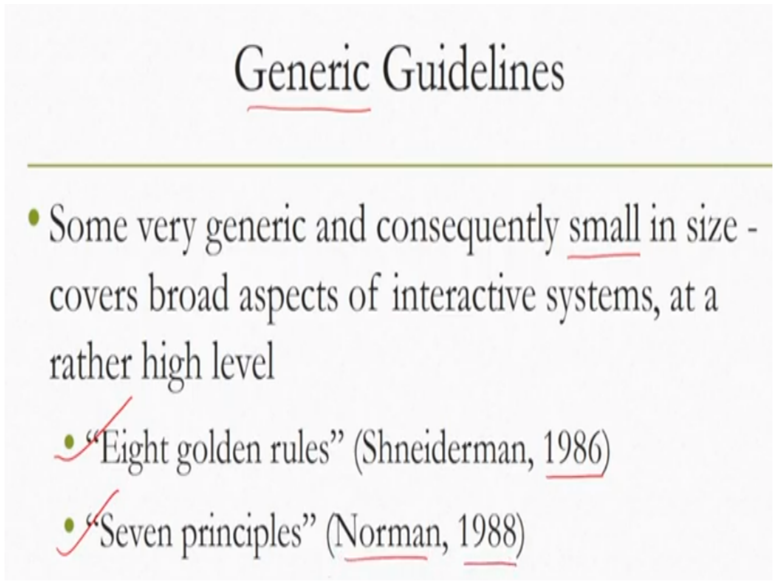


Guideline Types

- Many guidelines are there

So, there are different ways to look at the guidelines broadly there are 2 categories of guidelines.

(Refer Slide Time: 09:14)



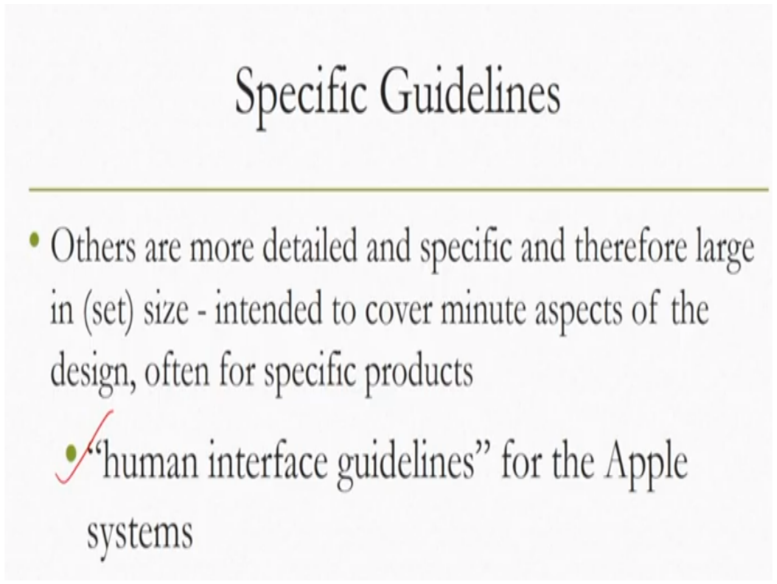
Generic Guidelines

- Some very generic and consequently small in size - covers broad aspects of interactive systems, at a rather high level
  - "Eight golden rules" (Shneiderman, 1986)
  - "Seven principles" (Norman, 1988)

Some are very generic in nature and since they are generic, they only refer to broader aspects of the design, rather than going into the minor details of specific system design. And accordingly, since they primarily refer to broader aspects of design which is generally true for any system rather than specific systems. Generally, these are at a very high level and the number of such guidelines is very less.

So, the overall set of guidelines is small we mentioned a couple of such guidelines such as the 8 golden, rules by Schneiderman proposed in 1986 and principles by Norman proposed in 1988.

(Refer Slide Time: 10:10)



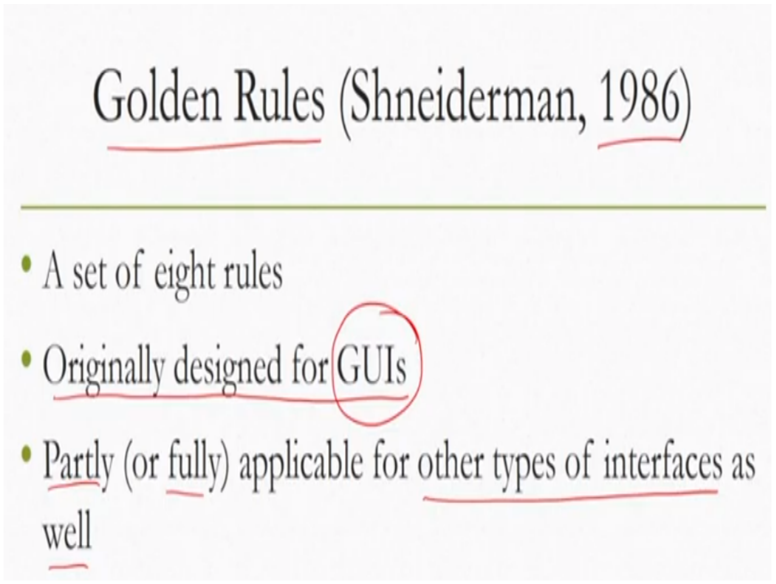
Specific Guidelines

- Others are more detailed and specific and therefore large in (set) size - intended to cover minute aspects of the design, often for specific products
  - "human interface guidelines" for the Apple systems

In contrast to these generic guidelines, we can have specific guidelines also where the guidelines are designed to build a specific type of system. So, the target is very specified and the guidelines are very minute in nature. So, they deal with intricate details of the overall system design. For example, human interface guidelines for the apple systems built for apple devices. So generally, these guidelines are very large in number and the set size is big.

So in this lecture, we are going to talk about one of the 2 generic guidelines, that we have mentioned earlier namely the 8 golden rules by Schneiderman's guidelines for the design of interfaces. In particular, these guidelines are applicable for the design of graphical user interfaces.

(Refer Slide Time: 10:49)



Golden Rules (Shneiderman, 1986)

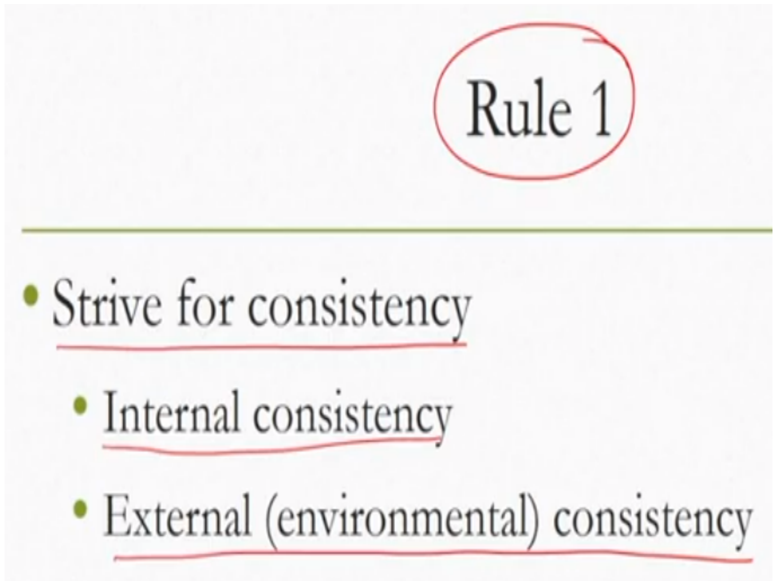
- A set of eight rules
- Originally designed for GUIs
- Partly (or fully) applicable for other types of interfaces as well

So, these are called golden rules so there are 8 golden rules as I said, they are originally designed for graphical user interfaces or GUIs. Now, the golden rules were proposed by Ben Schneiderman way back in 1986. At that time, the idea of the personal computer was coming into being and people were trying to develop graphical user interfaces for such computers.

So, it becomes popular and acceptable to the masses in that context, the golden rules were designed. However, as it belongs to a generic set of guidelines so, they broadly refer to larger and broader aspects of a design which makes them suitable partly or fully for other types of systems and interfaces as well. That is the advantage of having a generic guideline set although, it does not provide us with details of, what to do?

But it provides us, a very broad picture of what should be done, and since it provides a broad picture. Such guidelines are applicable for systems or interfaces other than the ones for which those were intended.

(Refer Slide Time: 12:50)



Rule 1

- Strive for consistency
  - Internal consistency
  - External (environmental) consistency

Now let us see what are these 8 guidelines or 8 golden rules. Rule number 1 the first rule first golden rule says that we should strive for consistency. Now, there are 2 types of consistencies from the point of view of this particular rule. One is internal consistency. Another one is external or environmental consistency. So, what this guideline tells us or what this golden rule tells us is that when we are trying to design an interface, we should strive for both internal as well as external consistency.

Now, what does it mean? Internal consistency means that whatever symbols metaphors, icons, texts, or tasks that we are designing for one part of the system should be consistently used in other parts of the system. For example, suppose we are designing a GUI with multiple windows, and to close windows, we are using a red circle with a cross inscribed inside it which is a typical way of denoting the close button.

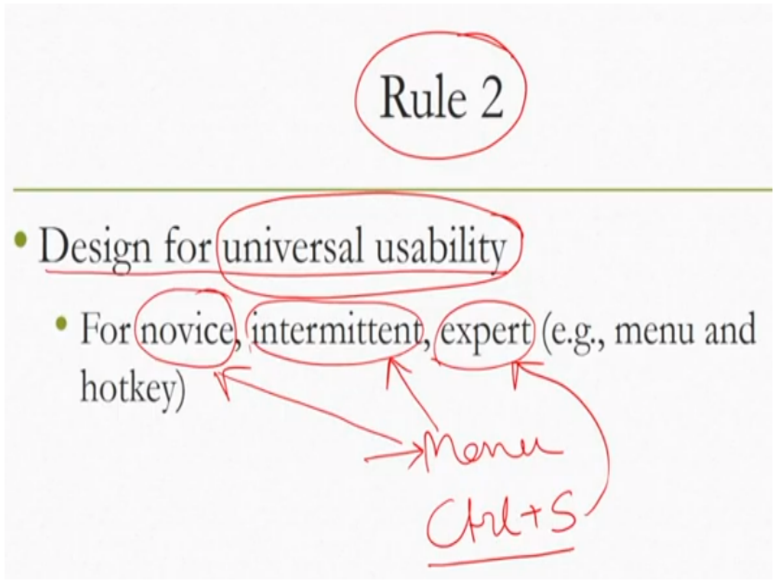
Now, for one window suppose we are using this red-filled circle with a cross and for another window, we are using a green-filled circle with a cross. Now, both windows are part of the same system. But, to close one window, we need to use the red circle with a cross, and to close another window, we need to use a green circle. So, there is a lack of consistency. If we want to perform the action of close, we should use consistent symbols or buttons or metaphors or icons to perform this operation across all the interfaces in the same system.

That is what is referred to as internal consistency same thing should be used everywhere with the same meaning. What is external consistency? Now, we may use some standards in our design. However, in our day-to-day life, we get to see something, we experience something so whatever we are defining for our system it ideally should not violate what we experience in our everyday situations. If these, are 2 matches, then we have external consistency. If these 2 do not match, then we do not have external consistency.

For example, we get to see traffic signals having 3 colours red to indicate stop, yellow to indicate slow down, and green to indicate go ahead. Now, suppose in our interface, we want to indicate stop or close and window. So ideally, we should use red colour to indicate this operation. Because stop or close in the context of a traffic light, which we experience in our day-to-day life is indicated by the red color.

So, in our system, if we are using red color, then we are consistent with our real-life experience. Now, instead of red, if we use say green to indicate close or stop. Then, of course, that is violating external consistency. So, you may note that we may maintain internal consistency but, violate external consistency which is quite possible. However; ideally, for an interface to be used, it is preferred that we follow both internal and external consistency which was rule one.

(Refer Slide Time: 17:24)



Rule 2

- Design for universal usability
  - For novice, intermittent, expert (e.g., menu and hotkey)

Now, let us talk about rule 2 this rule states that we should design for universal usability. Now, universal usability is a term, to be noted it indicates that our design should cater to different groups of users. Now, it should not be confused with the idea that the system that we design should be used by everybody. Now, that is something that violates the definition of usability as we have noted earlier.

Instead, what we are saying here is that? We should have a system, but for a specific group of users, a specific interface should be used. It is not that everybody gets to see everything and gets to use everything. Now, broadly we can categorize users as we have already seen in 3 groups novice, intermittent, and expert. So, novice users are those who typically use the system for the first time.

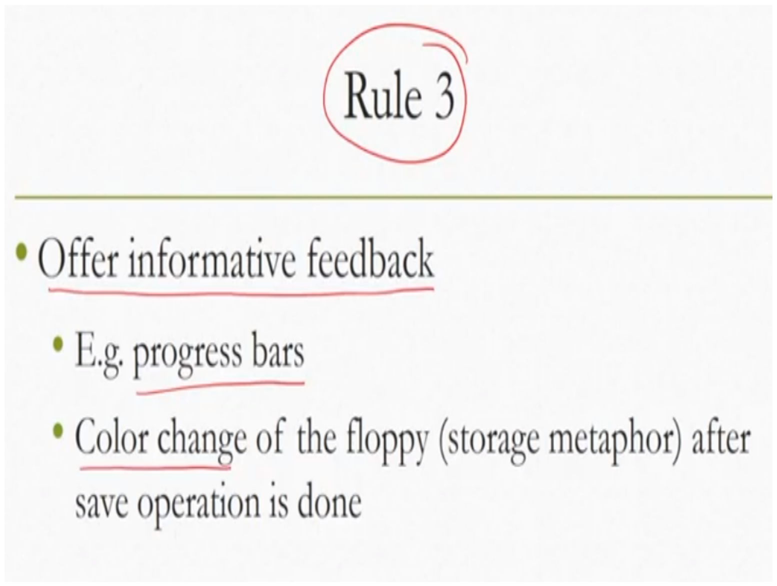
Intermittent users are those who occasionally use the system maybe in between there may be a long gap, and expert users are those who use the system regularly, and frequently. Now, for each category of users, we should have different features in the system. That is what is meant by design for universal usability. So, our system should be usable for novice users, should be used for intermittent users, and should be usable for expert users.

How by designing the features and functionalities provided in the system in a way. Such that, each group is scattered and the corresponding features are only visible to that particular group. All user groups should not be exposed to all features provided in the system. Then it affects usability. An example can be the use of a text editor so, all of us have used text editors such as ms-word or word pad.

Now, in such text editors suppose, I want to create a text and save it. How I can? Save so I can use either of 2 ways. One is using a menu-based option. So, in the menu there is a file menu typically under file there is a save option. So, I can select the file and the drop-down menu appears and then I can select the save option, to save the text. Alternatively, what I can do is I can use a combination of keys these are typically called hot keys.

For example, Ctrl +S is the most common hotkey combination used to save a file. So, I can either be using a menu or I can either use the Ctrl +S. Combination of these 2 keys control key and S key on the keyboard. Now, the menu-based option is typically good for novices and maybe also intermittent whereas Ctrl +S this hotkey combination may be good for experts. So, the system that we design should have facilities for both menu-based saving and Ctrl +S or hotkey-based saving, and the user depending on the category can use the specific option to perform the task. Then the interface will be usable to all the groups.

(Refer Slide Time: 21:10)



Rule 3

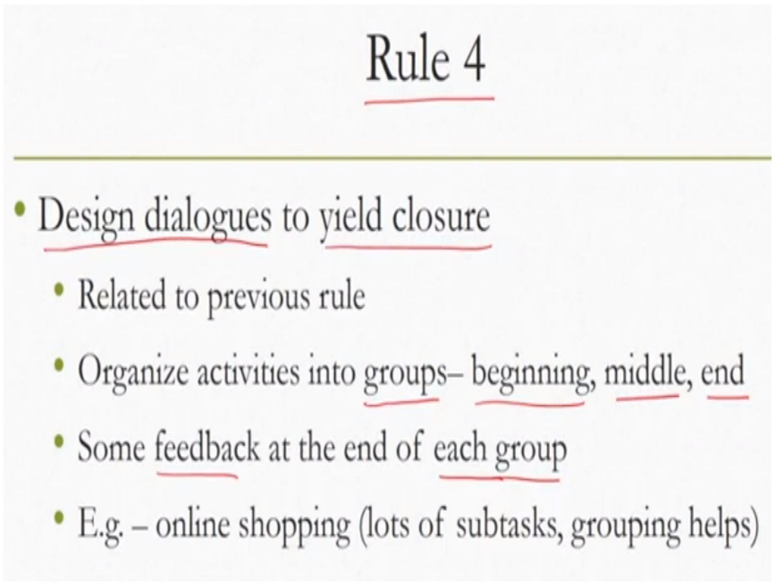
- Offer informative feedback
  - E.g. progress bars
  - Color change of the floppy (storage metaphor) after save operation is done

Now let us come to rule number 3 it talks about the fact that when we are designing something, we should offer informative feedback to users, whenever a user performs, some activities with the interface. An example is a progress bar. So, whenever the user performs some tasks and the progress of the task is shown in the form of a progress bar, then the user gets to understand how much of the task is done and how much is remaining?

This gives the user some sort of information about the completion time. Similarly, whether a task has been completed successfully or not can also be indicated to the user by some informative feedback. For example, a change in color on a menu option, so typically we use save menu option to save some editing task. And this menu option is accompanied by a metaphor of floppy disk, which is used to indicate storage device.

Now, before saving it comes with a color and after saving the color of the disk may change, it becomes grays which indicate that save option is done and then based on that change in color user may get to know, whether the save option is successfully done or not. If the color does not change, that means save is not done. So, we have to save again. Otherwise, if the color is changed that means the file is already saved. No need to do anything else. So, these are instances of informative feedback that should be provided to the user ideally.

(Refer Slide Time: 23:03)



Rule 4

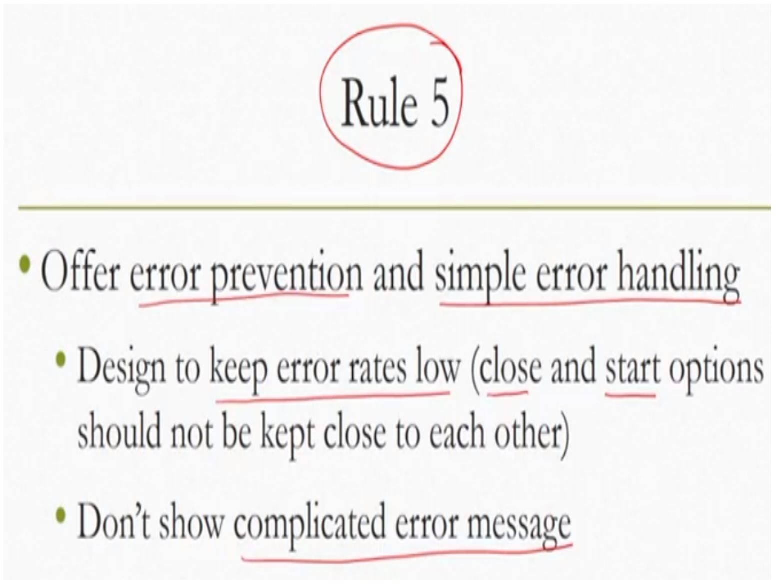
- Design dialogues to yield closure
  - Related to previous rule
  - Organize activities into groups– beginning, middle, end
  - Some feedback at the end of each group
  - E.g. – online shopping (lots of subtasks, grouping helps)

Rule 4 tells us, to design dialogues to yield closure of some operations. So, if the user again is performing some operation, then apart from providing feedback there should be some sort of dialogue between user and system, to take the user towards the closure of the operation to guide the user. So in order to do that ideally, we should try to organize any activity into one of the 3 groups beginning part, middle part and end part.

So, activities can be grouped into beginning activities, the middle part of the activities, and lending activities. Ideally, some feedback at the end of each group should be provided to indicate that a particular part of the activity is over. For example, when we are trying to purchase something online which; is a big complex task through some interface. Then, beginning of the purchase task actual purchase, and ending or checking out of the website, we can divide all the activities into these 3 categories.

And after each category, we should try to provide some hints to the user in the form of dialogues, to enable the user to understand the current stage where he or she is. This actually the dialogues complement the informative feedback.

(Refer Slide Time: 24:53)



Rule 5

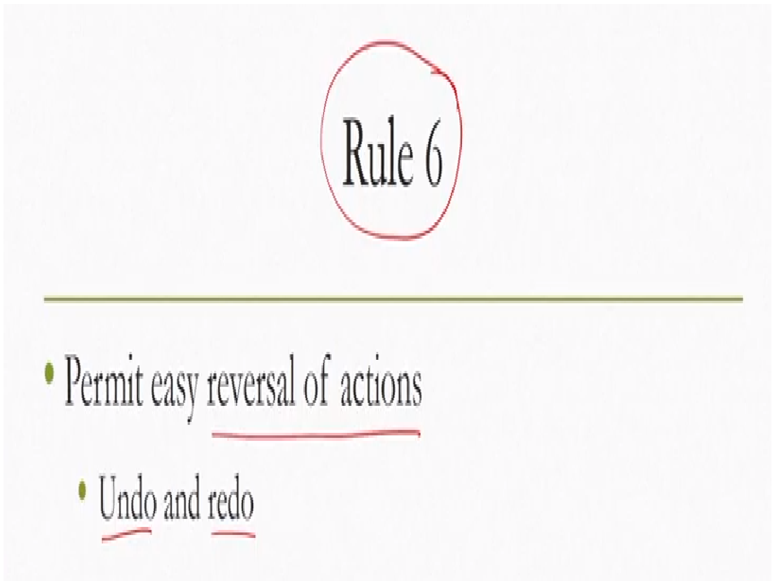
- Offer error prevention and simple error handling
  - Design to keep error rates low (close and start options should not be kept close to each other)
  - Don't show complicated error message

Then comes the fifth rule fifth golden rule of Schneiderman which says that ideally, designers should offer error prevention and simple error handling mechanisms in the design. So, any designer should strive for preventing errors by the users, and if prevention is not possible, which typically is the case then, how to come out of the erroneous situation that mechanism should be provided. The objective should be to keep error rates low and there should be some error handling mechanism.

For example, when we are performing some activity through some window the close and start options should not be kept closer to each other. In that case, erroneously user may select close instead of start or start instead of close which will increase the error rate. That is a very typical example of how to keep error rates low. Also, once some error is made error happened then, complicated error messages should not be shown to the users ideally.

So, the error messages should be in the understandable form in natural languages with as few technical terms as possible that should be the objective of any design of interfaces. So, rule 5 deals with, how to deal with errors.

(Refer Slide Time: 26:34)



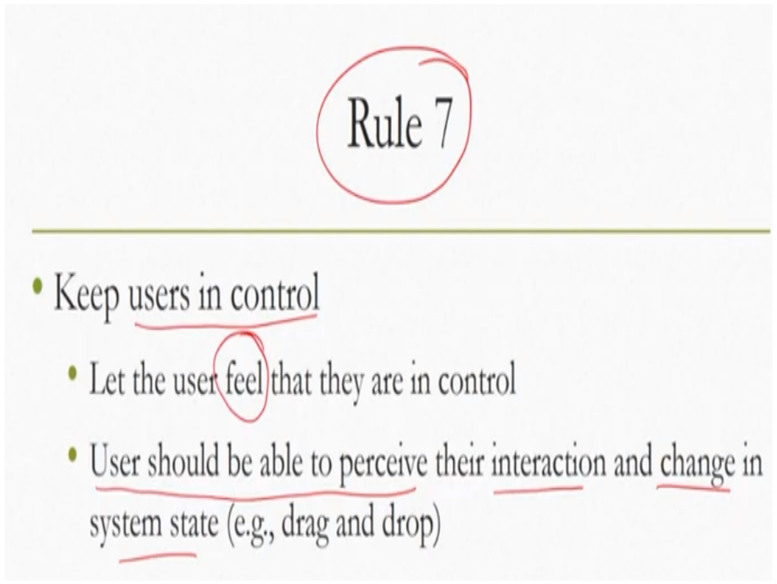
Rule 6

- Permit easy reversal of actions
  - Undo and redo

Rule 6 in continuation of rule 5, tells us to allow users to perform a reversal of actions. So, suppose we have made some mistakes or some errors happened, we want to get out of this. Now, get out and go where typically we should be able to reach a state where we are safe where things were not as bad as it is in the current state. So, that is called reversal of action. So, whatever actions we have done landed us into this current state, we want to reverse those actions.

We want to go back to an earlier state where such errors were not there. Rule 6 tells us that, our design should support such reversal of actions, and nowadays these are taken for granted in any GUIs or other interfaces you have heard of these terms undo operation and redo operation. These; actually allows us to reverse immediate actions that these; operations allow us to reverse actions that happened immediately before the last action.

(Refer Slide Time: 27:57)



Rule 7

- Keep users in control
  - Let the user feel that they are in control
  - User should be able to perceive their interaction and change in system state (e.g., drag and drop)

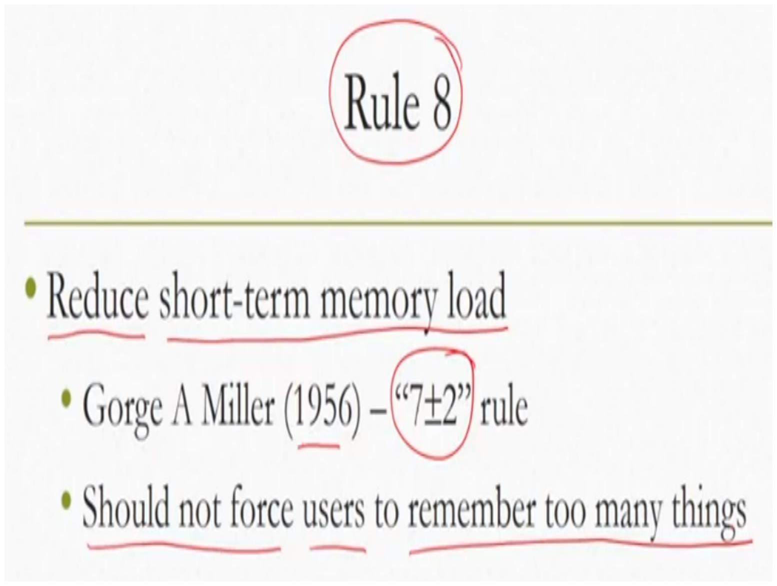
The seventh rule is number 7 it tells us that designers should strive to keep users in control or designers should strive to let the users feel that they are in control. So, this is very important that the users should feel that they are in control of whatever operations are being done on the interface. To get this feeling what is needed is that? The user should be able to perceive their interactions and change in the system state.

For example, suppose the user wants to move a file from one folder to another folder through command line if a move command is given then the system does the move operation. But the user does not know whether or does not feel whether the actual file has been moved or not. Instead, if the user can quote unquote, select a file representing it as a metaphor and drag it to a folder metaphor representing the storage and then release the selected file inside that folder quote unquote folder.

In other words, if the user is allowed toper form drag and drop operations to achieve the file movement task in the system. Then the user actually feels that he is able to quote-unquote see what is happening, he is able to select the file he wants, he is able to move it to the location he wants, and he is able to place it in the location he wants this gives the user a feeling of being in control of the system.

So, this type of visualization of the operations helps the user, feel in control and that should be the aim of any designer of interfaces.

(Refer Slide Time: 30:23)



Rule 8

- Reduce short-term memory load
  - Gorge A Miller (1956) – "7±2" rule
  - Should not force users to remember too many things

Finally comes, the last rule which is rule number 8. Now, this is a little different than the earlier rules. What it says is that the designer should try to reduce the short-term memory load of the user. Now, here it is assumed that a human being possesses 2 types of memory long-term memory and short-term memory. Of course, this is a very simplistic way of looking at our minds. But, from a practical point of view, it helps a lot.

So, when some interaction takes place, it is assumed that the knowledge relevant to that interaction is loaded into the short-term memory and from there we make use of that knowledge and perform the interaction. Now, if the interaction requires too much knowledge, then it creates a problem. Because short-term memory is supposed to have a limited capacity it cannot contain an infinite amount of knowledge.

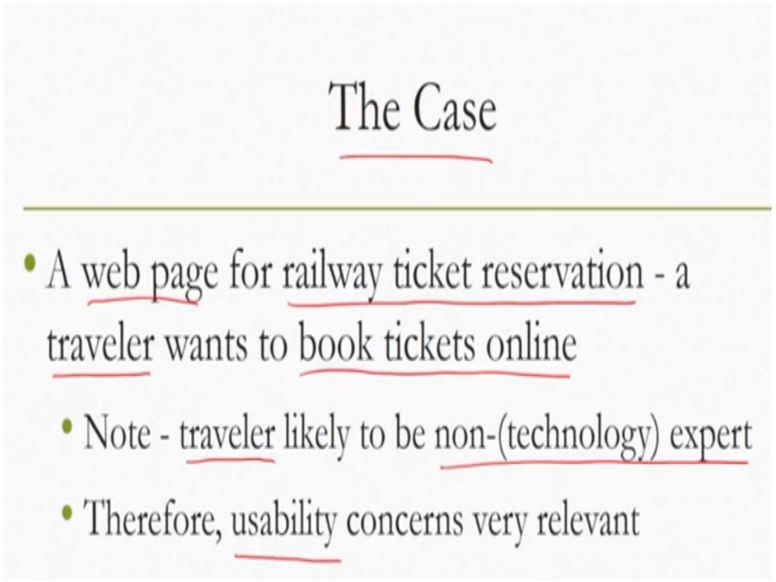
So, within the limited capacity if the knowledge is restricted then interaction takes place smoothly and usability improves. However, if the knowledge required to interact is more than the capacity, then it affects usability and the interaction tends to be not so smooth. So, there is one theory by George miller, which was proposed way back in 1956 says that it is known as the 7 plus minus 2 rule which says that our short-term memory can at a time hold between 5 to 9 pieces of information.

Now, these pieces of information of course are not clearly defined. But we can assume them to be units of information that our short-term memory can hold between 5 to 9 units and it varies from person to person. So, average number of pieces of information that can be stored in 7 varies from 5 to 9 from 1% to another. So, what this 7 plus minus 2 rule, tells us is that a design should not force users to remember too many things.

Suppose, somebody can hold seven units of information, then if a design requires the user to make use of more than seven units of information then of course that interaction is likely to fail. Because the short-term memory; capacity is exceeded and the user will not be able to bring in that knowledge to operate the interface. So, we should be very careful while designing our interface. We should not force the user to remember too many things which in turn is going to affect the interaction.

So, these are the 8 golden rules proposed by Schneiderman and while going for any interface design, we should take these rules as a starting point. Whatever our intuition tells us about the design we should test it with respect to these rules and see whether these rules are violated if they are violated, then we should refine our design and come up with a modified design. So, that is how we should proceed.

(Refer Slide Time: 34:12)

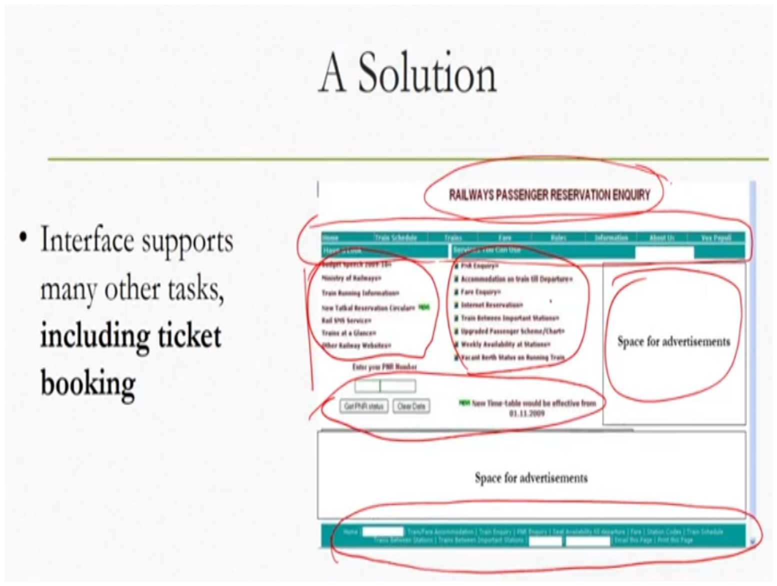


The Case

- A web page for railway ticket reservation - a traveler wants to book tickets online
  - Note - traveler likely to be non-(technology) expert
  - Therefore, usability concerns very relevant

Now, let us try to understand the importance of these rules in the context of a system design let us try to understand it with respect to a case study. So, what is the case, the case here is a web page for railway ticket reservation, to be used by a traveler. So, the traveler wants to book tickets online through the web page and here of course we are assuming that the traveler is a non-technology expert or in other words layman user of the interface. So, the interactive system principles apply and accordingly usability concerns are very relevant in this case.

(Refer Slide Time: 35:07)



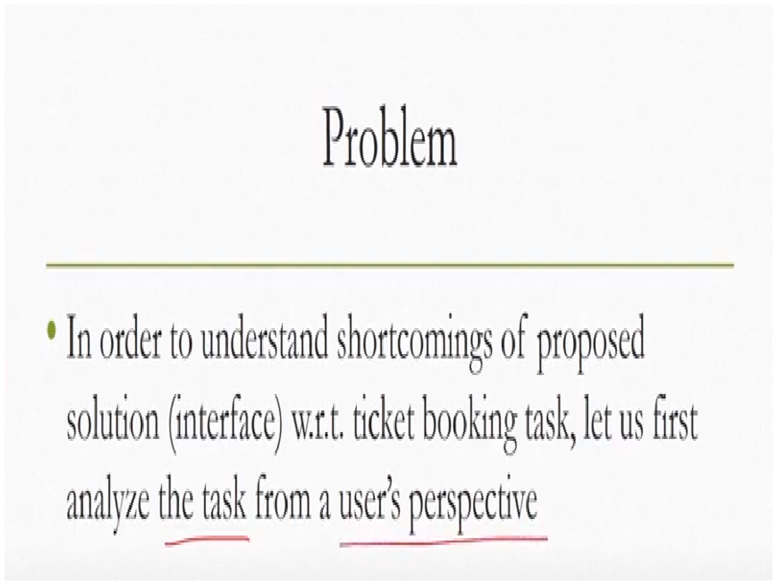
A Solution

- Interface supports many other tasks, **including ticket booking**

Now given this problem, we need a web page that allows a traveler to book tickets for traveling by train. Suppose somebody proposed a design of an interface for railway passenger reservation inquiry. It contains several option there is this top-level menu containing several options then this middle part has several sets of menus, left side, central side and there is some other web real estate used for some other purpose.

At the bottom there are some options again there is another menu at the very bottom of the interface. So, each of these options is essentially hyperlinks after clicking we get to see a different page.

(Refer Slide Time: 36:17)

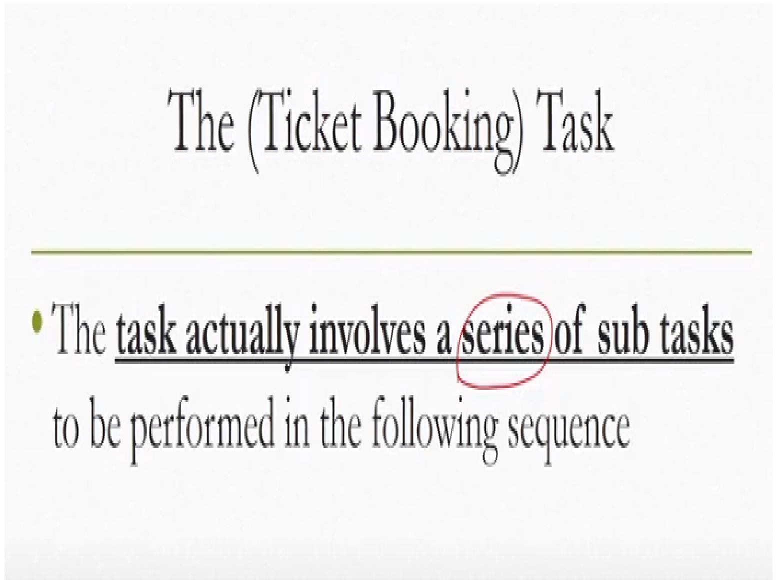


Problem

- In order to understand shortcomings of proposed solution (interface) w.r.t. ticket booking task, let us first analyze the task from a user's perspective

So, whether it serves our purpose it can serve definitely. But what are the problems are there any problems with this interface? Let us try to understand with respect to the task that is booking of a ticket by a traveler. So, let us first try to analyze the task from the point of view of a traveler who is the user of this interface.

(Refer slide Time: 36:47)

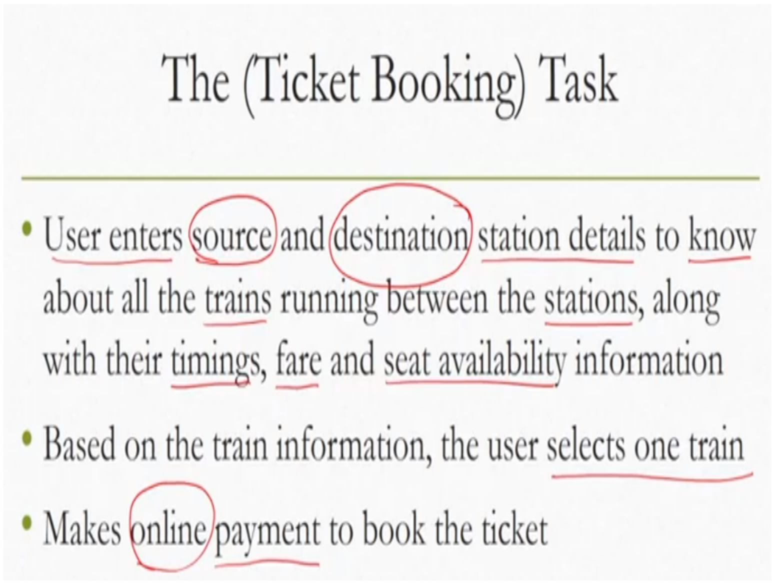


The (Ticket Booking) Task

- The **task actually involves a series of sub tasks** to be performed in the following sequence

Now, the ticket booking task actually involves a series of sub-tasks. So, it is not a single task it can be broken down into sub-tasks. In sequence what is the sequence?

(Refer Slide Time: 37:04)

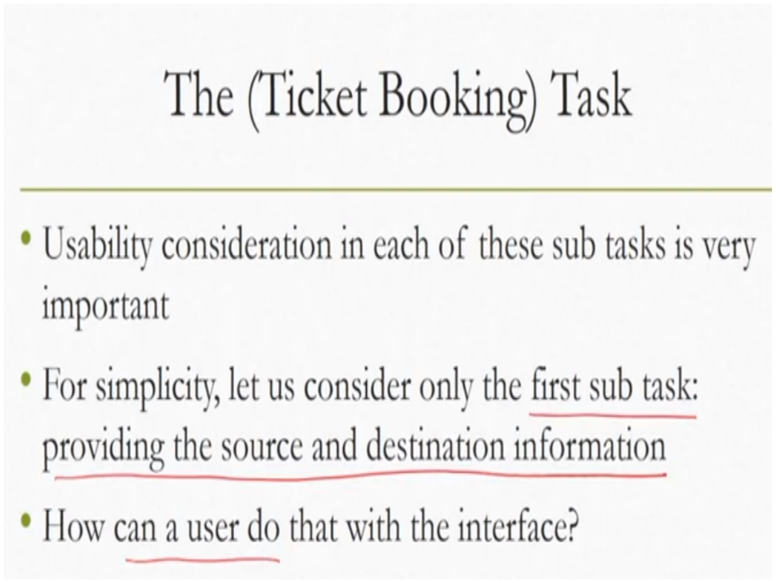

The (Ticket Booking) Task

- User enters source and destination station details to know about all the trains running between the stations, along with their timings, fare and seat availability information
- Based on the train information, the user selects one train
- Makes online payment to book the ticket

The first user enters the source, station details, and destination station details. To know about all the trains that is running between the stations. That is the very first information that somebody needs to know before he or she proceeds to book tickets. Not only the trend names along with their timings ticket fare and availability of seat all this information is required before someone proceeds for booking of tickets.

So, the first task is to get that information in the second subtask, based on the train information the user selects one train. So now, the second subtask is a selection of the train along with a selection of seats. And finally, some payment is made to book the tickets so payment is made online to book the ticket because everything is being done online. So broadly there are these 3 sub tasks one is first to get train information then select a train with seats and finally make payment.

(Refer Slide Time: 38:35)

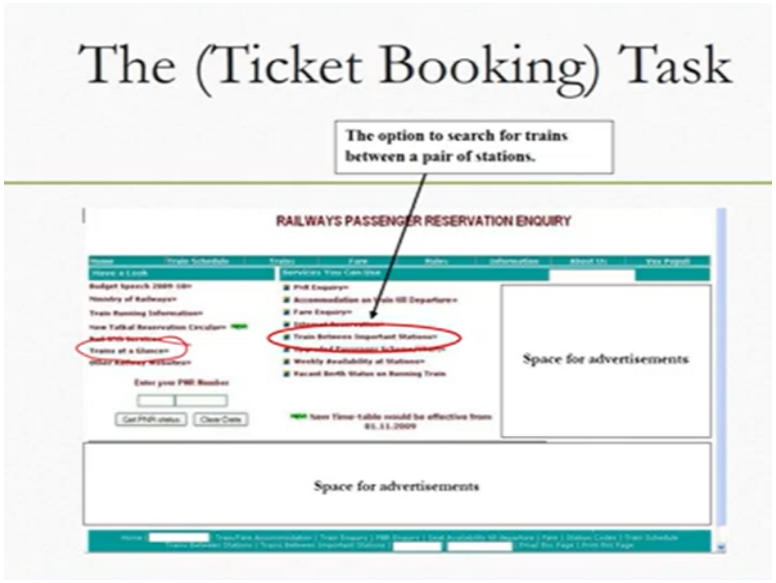


The (Ticket Booking) Task

- Usability consideration in each of these sub tasks is very important
- For simplicity, let us consider only the first sub task: providing the source and destination information
- How can a user do that with the interface?

If we see carefully, we will see that each of these sub tasks can further be broken down into sub tasks and usability concerns for each of these series of sub-tasks are there. For simplicity, let us consider only the first sub-task that is providing the source and destination information. How do we provide the information to get the train details using the interface? So, with that interface how can a user provide such information? Let us see let us have a closer look at the interface to find out how a user can do that.

(Refer To Slide Time: 39:23)



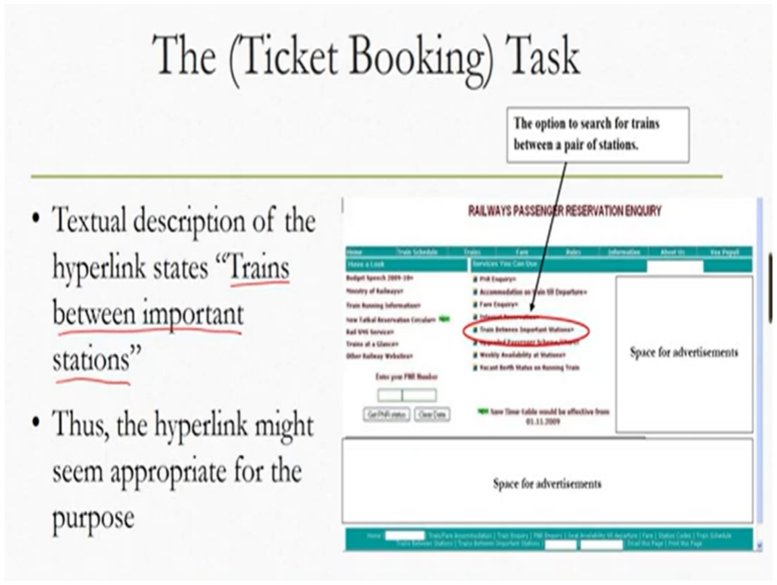
The (Ticket Booking) Task

The option to search for trains between a pair of stations.

So, this is the interface again, now in the interface, there is a hyperlink as marked here. The hyperlink says that trains between important stations. So, this is all the menu options or hyperlinks that relate to something providing station details. All other options if you check have no apparent indication that through those links, we can provide station details. For example, there are other hyperlink trends at a glance.

But, as the name suggests it provides information about all trends running everywhere in the geographic location, rather than allowing us to provide some station names to get specific trend details.

(Refer Slide Time: 40:21)



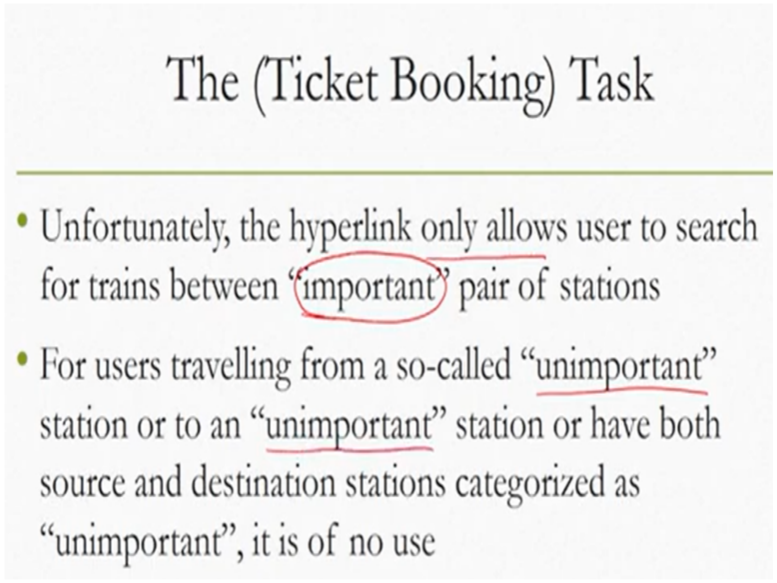
The (Ticket Booking) Task

- Textual description of the hyperlink states "Trains between important stations"
- Thus, the hyperlink might seem appropriate for the purpose

The option to search for trains between a pair of stations.

Now, the textual description says trends between important stations. This option seems to be the closest to what the user wants to achieve. So, in order to use this hyperlink, what the user needs to do?

(Refer Slide Time: 40:40)

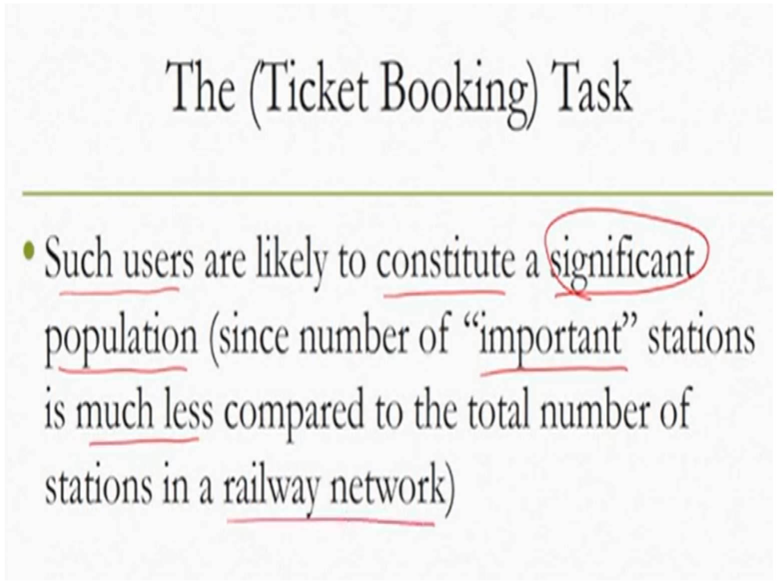


The (Ticket Booking) Task

- Unfortunately, the hyperlink only allows user to search for trains between "important" pair of stations
- For users travelling from a so-called "unimportant" station or to an "unimportant" station or have both source and destination stations categorized as "unimportant", it is of no use

As the name suggests the hyperlink only allows users to search for trends between important pairs of stations. Now, it may so happen that the station pair that the user is interested in does not quote-unquote important. In that case, of course, there is no way out. So, if a traveler is traveling or willing to travel from one unimportant station to another unimportant station then this particular option is of no use.

(Refer Slide Time: 41:14)

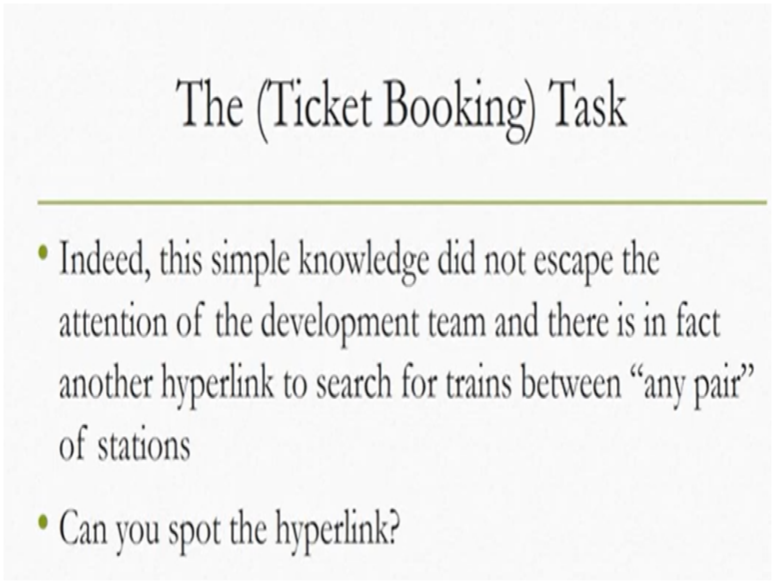


The (Ticket Booking) Task

- Such users are likely to constitute a significant population (since number of "important" stations is much less compared to the total number of stations in a railway network)

In fact, such users are likely to constitute a significant population of the overall user population. Because the number of important; stations is generally much less compared to the total number of stations in a railway network.

(Refer Slide Time: 41:38)

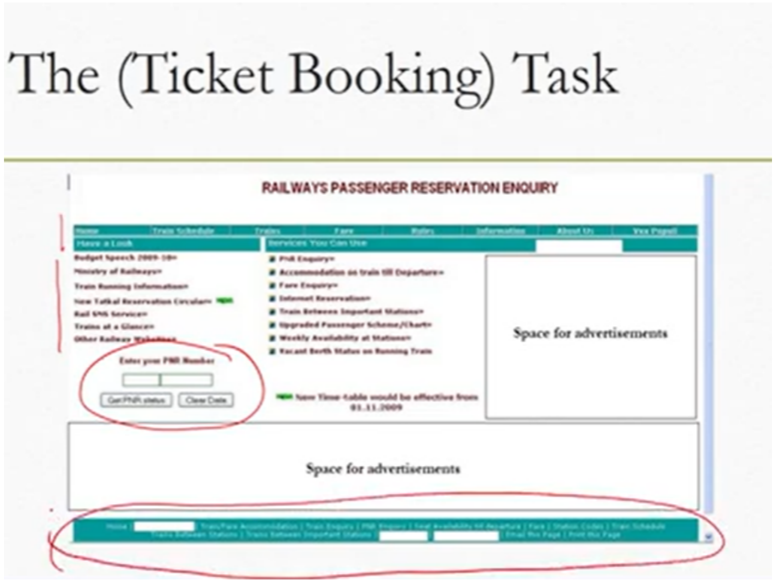


The (Ticket Booking) Task

- Indeed, this simple knowledge did not escape the attention of the development team and there is in fact another hyperlink to search for trains between "any pair" of stations
- Can you spot the hyperlink?

Now, then appeared with a cursory look at the interface it may appear that for such users there is no option. But, that is not true the designers also have probably thought of this issue and provided a way out for such users. So, there is another hyperlink available that can allow such user's to perform the task. So, on the interface, it is very difficult to find out that hyperlink.

(Refer Slide Time: 42:11)



The (Ticket Booking) Task

RAILWAYS PASSENGER RESERVATION ENQUIRY

Home | Train Schedule | Trains | Fare | Rules | Information | About Us | Yes Pagell

Have a Look

- Budget Speech 2009-10»
- Ministry of Railways»
- Train Running Information»
- New Tatkal Reservation Circular»
- Rail SMS Service»
- Trains at a Glance»
- Other Railway Websites

Enter your PNR Number

Get PNR status | Clear Data

Services You Can Use

- PNR Enquiry»
- Accommodation on train till Departure»
- Fare Enquiry»
- Internet Reservation»
- Train Between Important Stations»
- Upgraded Passenger Scheme/Chart»
- Weekly Availability at Stations»
- Vacant Berth Status on Running Train

New Time-table would be effective from 01.11.2009

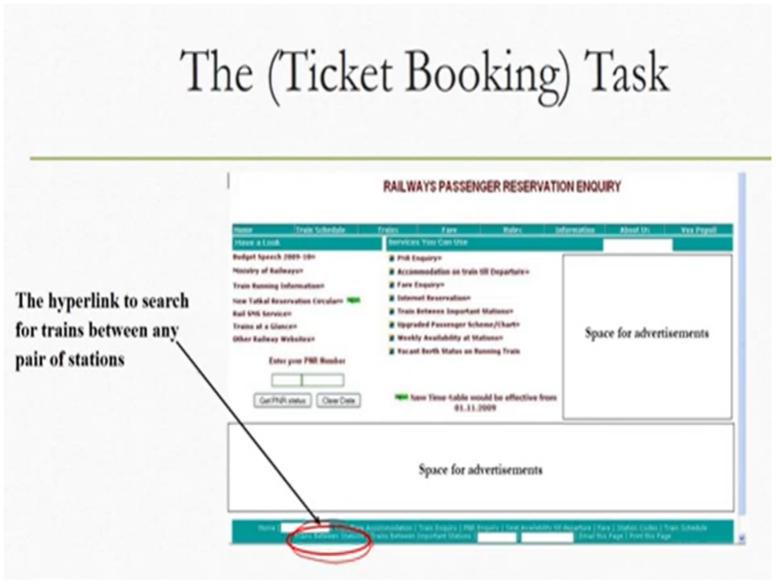
Space for advertisements

Space for advertisements

Home | [illegible] | Train/Fare Accommodation | Train Enquiry | PNR Enquiry | Seat Availability till departure | Fare | Station Codes | Train Schedule | Trains Between Stations | Trains Between Important Stations | [illegible] | Email this Page | Print this Page

Let us see the interface again. If we give a cursory loop can we get it now one thing we should keep in mind is that, if an interface is provided and it takes the user a lot of time to browse through lots of options and find out the one he or she is looking for then definitely that is not a good interface? Because it violates certain guidelines that we will see later here you can see there are lots of options. In the top-level menu, there are lots of options, in the middle-level menu, there is an even larger number of options than there are these things and the bottom-level menus.

(Refer Slide Time: 43:11)

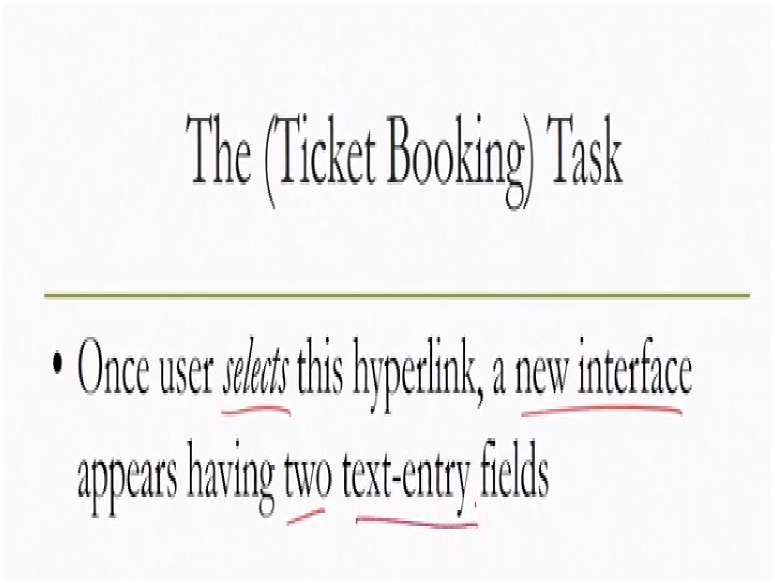

The (Ticket Booking) Task

The hyperlink to search for trains between any pair of stations

So among these options, it is really difficult to spot such a hyperlink, which allows users to provide details of any stations not only important stations where they can be. In fact, it is there, at the very bottom of the screen it says train between stations as highlighted here in this image. Now, see a cursory glance at the design tells us that the placement of this particular option actually deprives a significant segment of the user population of easily finding out the option.

Maybe after some usage, it may become easier but for a novice user, it really creates some problems. So, then if it is not very useful to an office user then that violates a guideline that is it affects universal usability. Let us see what else we can find out.

(Refer To Slide Time: 44:07)

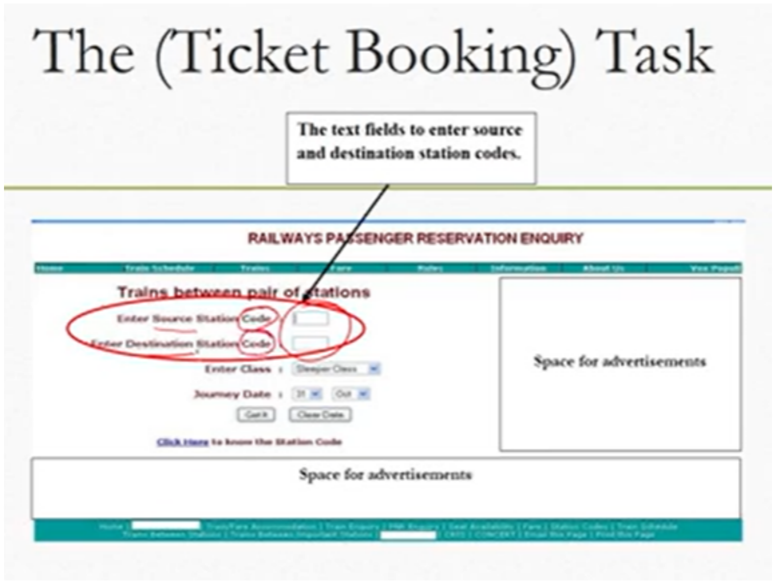


The (Ticket Booking) Task

- Once user selects this hyperlink, a new interface appears having two text-entry fields

Once users select this hyperlink since it is a hyperlink, a new interface appears. Now, in this new interface, there are 2 text entry fields.

(Refer Slide Time: 44:21)



The (Ticket Booking) Task

The text fields to enter source and destination station codes.

RAILWAYS PASSENGER RESERVATION ENQUIRY

Trains between pair of stations

Enter Source Station Code

Enter Destination Station Code

Enter Class : Sleeper Class

Journey Date : 31 Oct

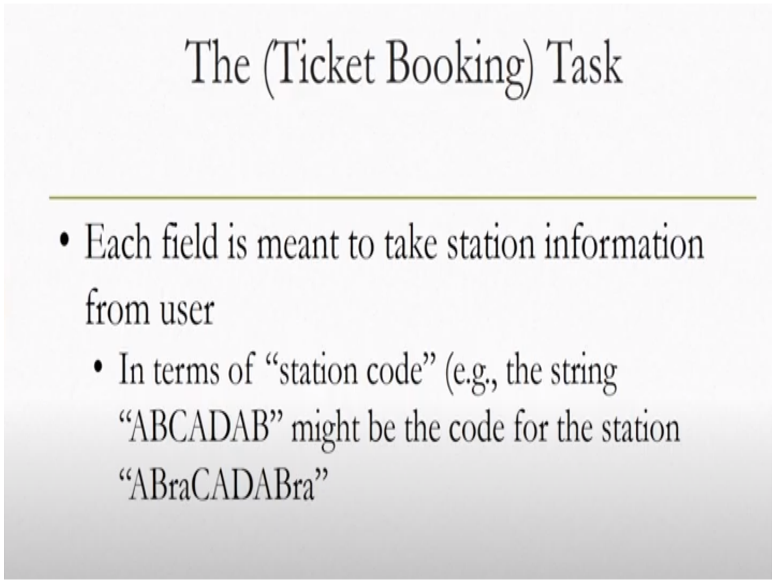
Click Here to know the Station Code

Space for advertisements

Space for advertisements

The interface looks something like this where these are the 2 text entry fields and here each field is meant to provide information about one station. So, the fast field is for the source station and the second field is for the destination station. But here it is not asking for names instead it is asking for codes of a station.

(Refer Slide Time: 44:50)

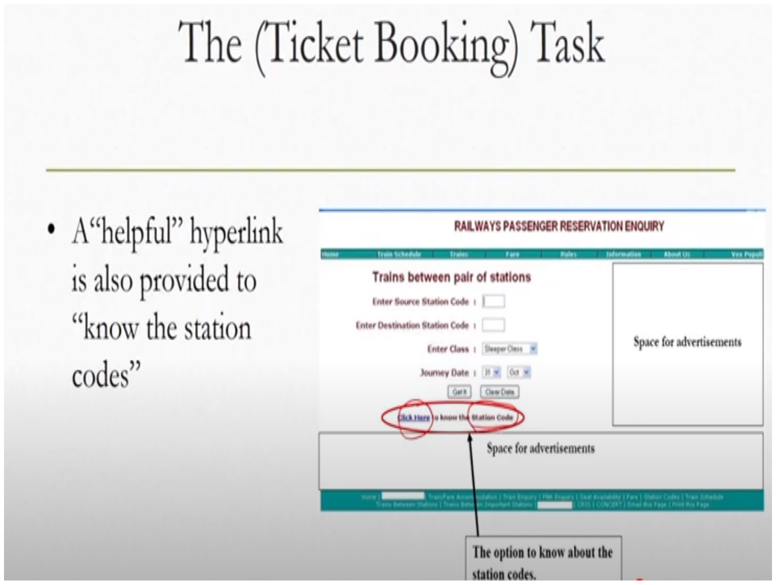


The (Ticket Booking) Task

- Each field is meant to take station information from user
  - In terms of "station code" (e.g., the string "ABCADAB" might be the code for the station "ABraCADABra"

Now, the code and name are different. So, the name may be very easy to remember. However, codes need not be easy to remember and codes are a typically shorter form of the name if you are using station code names you may be aware of which is not very easy to remember.

(Refer Slide Time: 45:17)



The (Ticket Booking) Task

- A "helpful" hyperlink is also provided to "know the station codes"

RAILWAYS PASSENGER RESERVATION ENQUIRY

Trains between pair of stations

Enter Source Station Code :

Enter Destination Station Code :

Enter Class : Sleeper Class

Journey Date : 31 Oct

Click Here to know the Station Code

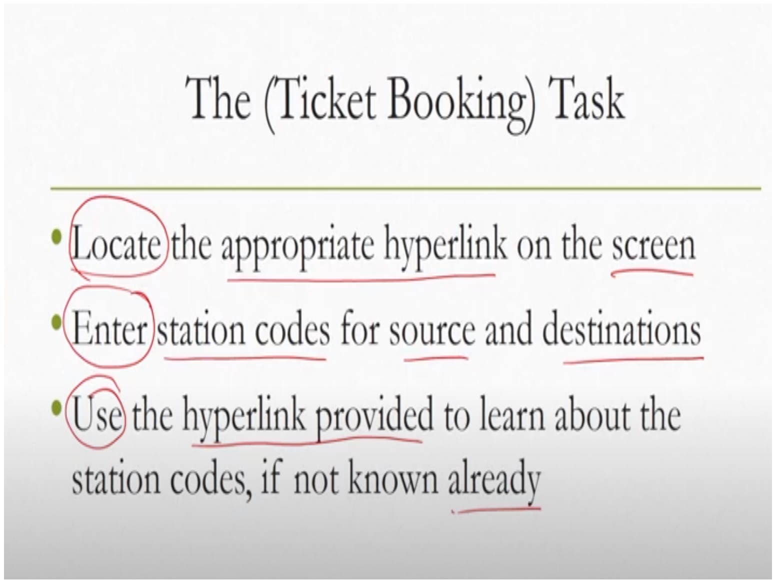
Space for advertisements

Space for advertisements

The option to know about the station codes.

Then what can be done? So, for those who cannot use the code names or cannot remember the codenames for them a link is provided where after clicking on this link a page appears where we can get to know the station code for a given station name.

(Refer Slide Time: 46:00)



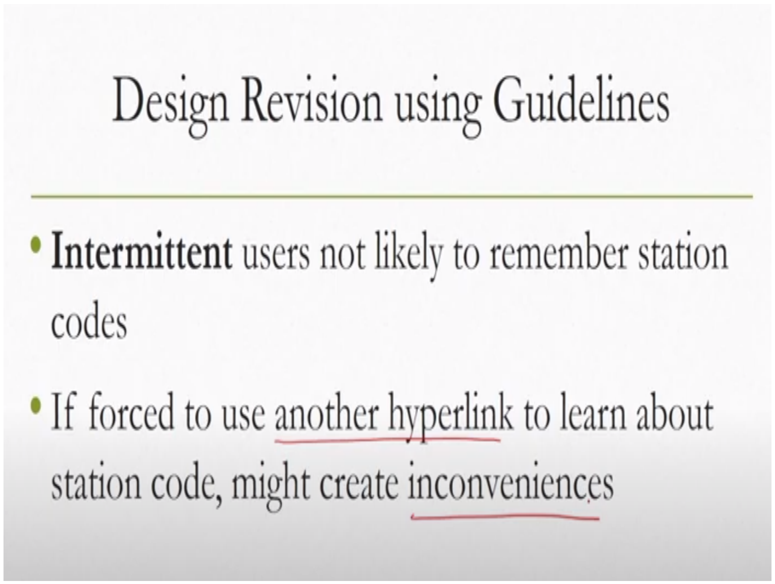
The (Ticket Booking) Task

- Locate the appropriate hyperlink on the screen
- Enter station codes for source and destinations
- Use the hyperlink provided to learn about the station codes, if not known already

So, in summary, to achieve the goal to perform the task of entering source and destination station names what the user gets to do. First of all the user needs to locate the appropriate hyperlink on the screen, which as I pointed out is not very easy for novice or intermittent users, it may be okay with expert users. Then enter station codes for source and destination and use the hyperlink provided on the interface to learn about station codes if not known already.

Now, this is related to only the first task which is to input the source and destination station name. Just to do that so many things the user needs to perform on the interface.

(Refer Slide Time: 46:49)

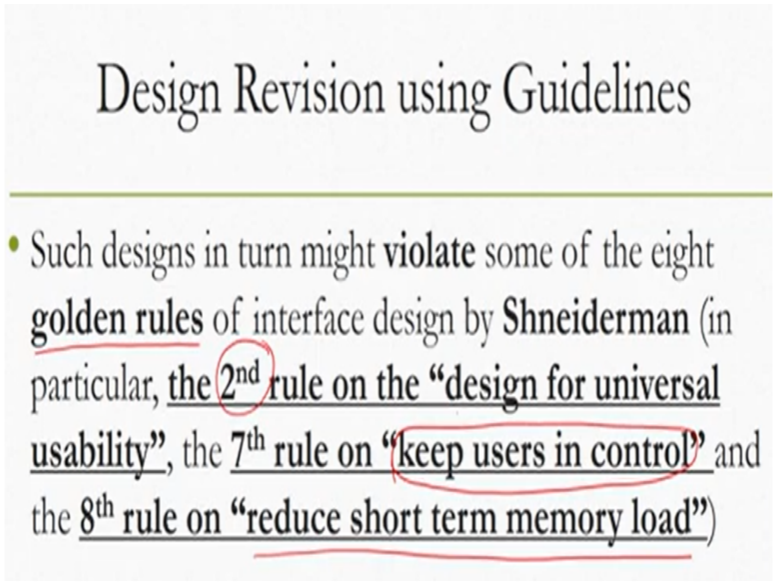


Design Revision using Guidelines

- **Intermittent** users not likely to remember station codes
- If forced to use another hyperlink to learn about station code, might create inconveniences

Now, many users significant user population which includes intermittent users, and no-vice users which may not remember which is quite natural remember codes for all the stations. Now, in that case, they are forced to use another hyperlink. Now, that actually creates inconvenience. Because again there is a screen change and too many screen changes create inconvenience for lay-persons lemon users.

(Refer Slide Time: 47:29)



Design Revision using Guidelines

- Such designs in turn might **violate** some of the eight **golden rules** of interface design by **Shneiderman** (in particular, **the 2nd rule on the "design for universal usability"**, the **7th rule on "keep users in control"** and the **8th rule on "reduce short term memory load"**)

So, these design decisions were just to enter 2 station names either the user has to remember the code or has to use another interface to learn the code or to place the option to enter station names or station codes. In an obscure position on the screen which makes it is difficult to find out the link in the very first place. These types of design decisions are not very good decisions they actually violate some of the golden rules. So, if the rules were used then probably such decisions would not have been made.

In particular, the design that we have just discussed violets the second rule design for universal usability, we have already seen how so effectively the design by keeping the link in a very obscure place makes it very difficult for novice and intermittent users to locate them in the first place. So, whenever they use it they find it difficult to find the specific hyperlink to locate the specific hyperlink.

So, it is not very useful to novice and intermittent users for frequent users this may not be an issue. Secondly, it violates the seventh rule which is to keep users in control. So, when we are asking users to enter station codes and if the station code is not known then forcing the user to select another interface then from there find out station codes. This actually makes the user feel that they are not in control of the overall operations on the interface.

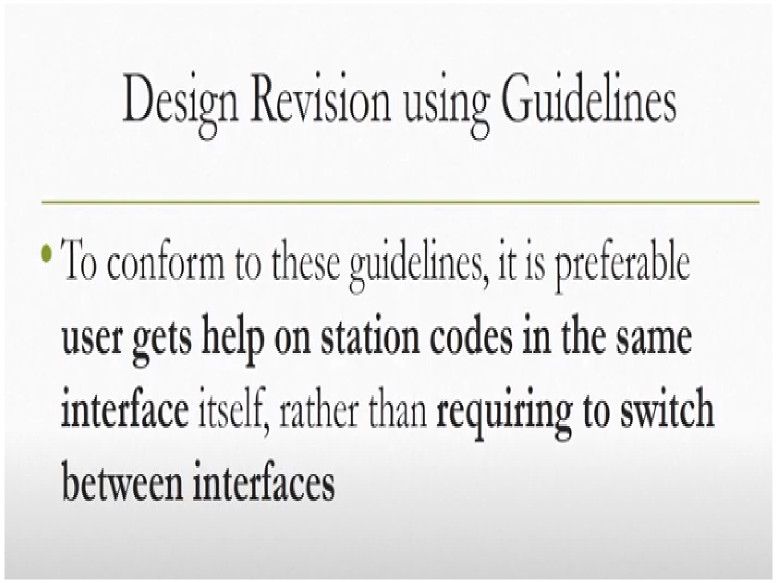
If the users would have been able to simply enter the station names instead of doing so many things then they would have probably felt it to be in control. However, the very design that was conceptualized and proposed makes it difficult for users to feel in control. So, that violates the seventh rule. It also violates the eighth rule which is to reduce short-term memory load. So, when we are asking the users to enter station codes, there can be hundreds and thousands of unimportant stations.

Users may like to choose any pair of stations. So, hundreds of thousands of codes need to be remembered but all those things will not be possible to accommodate in the short-term memory .Because of its limitations, it can hold maximum it can hold for some users 5 pieces of information for other users 9 pieces of information on an average 7 pieces of information. Each code we consider to be a piece of information then on average 7 codes is fine but that is not practical.

There will be hundreds and thousands of such codes and the interface forces the user to remember them otherwise the user has to do additional operations to know it. So, the user either has to remember or so either has to put more cognitive effort into remembering the codes or put more physical effort into selecting the other hyperlink going through the link, going through the information on the page and learn the code.

Both affect user's cognitive effort and in turn, it affects the eighth rule which is to reduce short-term memory load. So, these 3 rules are clearly violated in the particular design. That we have seen.

(Refer Slide Time: 51:34)



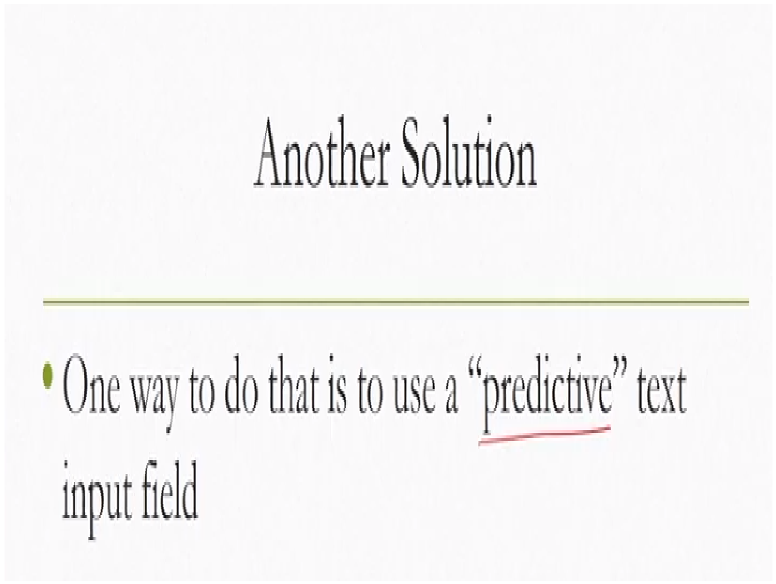
Design Revision using Guidelines

- To conform to these guidelines, it is preferable **user gets help on station codes in the same interface** itself, rather than **requiring to switch between interfaces**

Now, the design is done and we have identified that there are these 3 issues with respect to the 8 golden rules. So, what to do? Design for universal usability requires us to put this particular hyperlink of getting to input station details for any pair of stations on a prominent place preferably in the central place which is a design modification required. The second thing is once this hyperlink is selected and this second page appears where station codes need to be entered.

So instead of forcing the user to move to a third page, to get to know the station code within the second page itself, there should be some support provided. So, that user does not need to remember the code instead the system helps the user remember the code how that is possible?

(Refer Slide Time: 52:39)

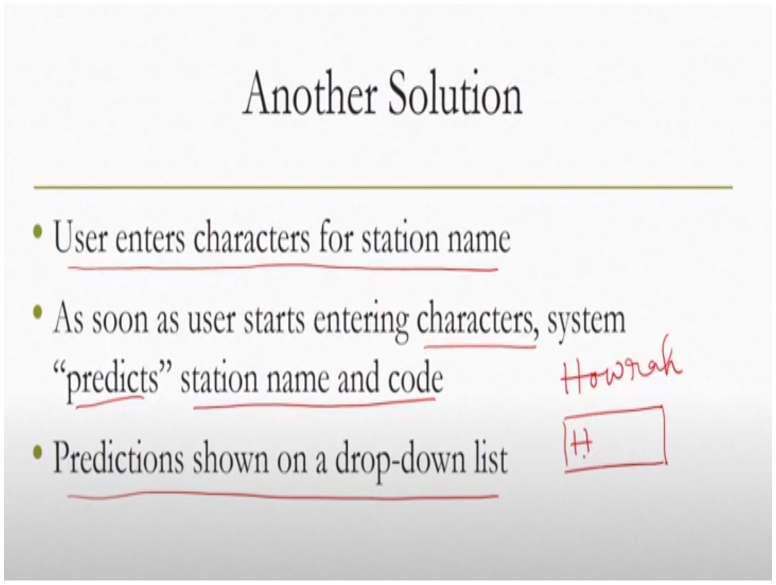


Another Solution

- One way to do that is to use a "predictive" text input field

One way to do that is to use a predictive interface. Where we can predict the text from a few characters? So, in this case, suppose the user does not know the station code. Now user remembers the station name, so from the name few characters of the name, once the user inputs then the code appear automatically by prediction, which will help the user in a great way.

(Refer Slide Time: 53:05)



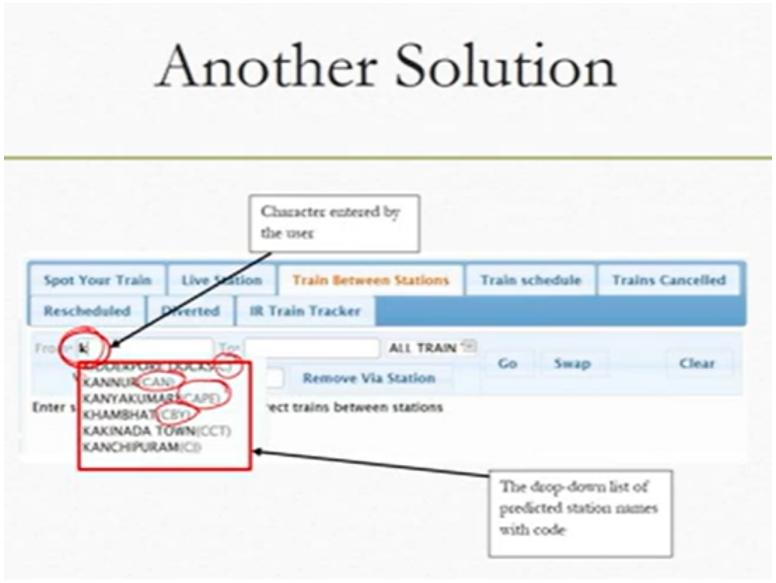
Another Solution

- User enters characters for station name
- As soon as user starts entering characters, system "predicts" station name and code
- Predictions shown on a drop-down list

So, the user enters characters for the station name. As soon as the user starts entering characters system predicts station name and code both so, that the user gets to understand that this code is meant for this name and the prediction can appear on a drop-down list. So, suppose this is the text field. Suppose I want to know the code for the station, Howrah the moment, I enter h than in a drop-down list, it shows codes corresponding to the station starting with the name H. So, there may be a station name and corresponding code.

Then I enter W so to refine the list enter W then this display changes to all the station names having HO as starting 2 letters and the corresponding codes. As soon as I locate the code, I just select it rather than, having to find it from a big repository of information. So, this type of predictive text interface helps us to effectively nullify the shortcomings that we have identified.

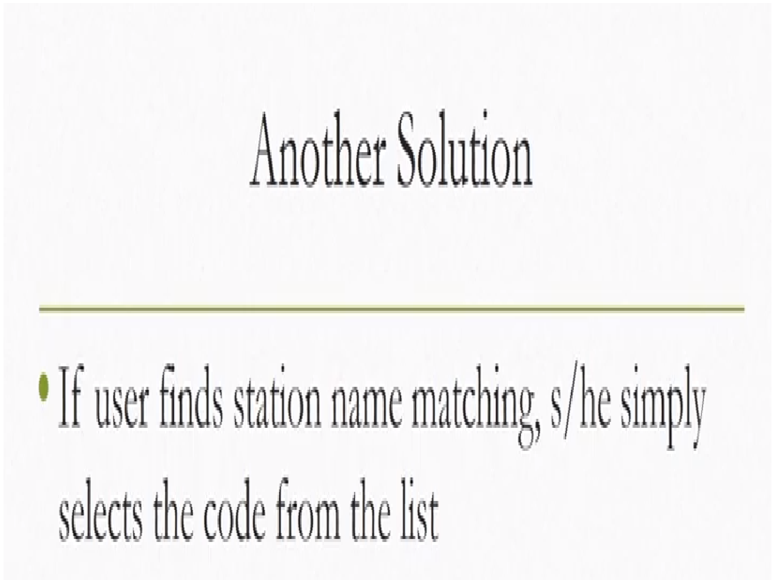
(Refer Slide Time: 54:41)



Another Solution

So, this can be one way of doing things. So, here the user entered K corresponding to K all the station names appeared, and beside the name, this code appears as shown here. So here then the user does not need to select the hyperlink and open another page to browse through the name of stations to learn about their corresponding code. Instead, the user can simply start entering the station name and the code automatically appears through prediction on a drop-down list and the user just need to select it. So, this alleviates the issue of violation of the golden rules.

(Refer Slide Time: 55:32)



Another Solution

- If user finds station name matching, s/he simply selects the code from the list

So, that is in summary what we can do with the guidelines. So, guidelines just to recollect. Guidelines are starting points. Now, we may be having some intuition about design and interface design. So, we can always use intuition to come up with the design but then whether the design is going to be usable or not, one way to do it is by going through the entire design prototype evaluation cycle. But before that also our intuition can be guided by the guidelines to come up with initial designs or if we have some initial designs, we can simply apply the guidelines to refine the design before we prototype and evaluate it.

So, in that way guidelines provide us with starting points in the design prototype to evaluate the life cycle .So, here in this lecture, we learned about the 8 golden rules of Ben Schneiderman which is a very generic set of guidelines applicable primarily for graphical user interfaces. But because of its generic nature can be applied to other types of interfaces as well. In the next lecture we are going to learn about another set of guidelines namely the 7 principles of Donald Norman which again was proposed almost at the same time, when the eight golden rules were proposed.

we will learn in more detail about those principles and what they refer to in the next lecture whatever we have discussed today can be found in this book particularly chapter 2 section 2.4.4 so that is all for this lecture I hope you enjoyed the learning and got to know about the guidelines and how it can be used in practice hope to see you all in the next lecture thank you and goodbye.