SLIIT

Discover Your Future

Lecture 01
INTRODUCTION TO ITP

IT2080 IT Project

B. Sc. Special Honors in Information Technology

Year 2 – Semester 2

Agenda

1. Introduction to ITP
2. Project Initiation
3. Guidelines to Succeed the Project
4. Assignment 1 - Project Proposal
5. Sample Case Studies

1. Introduction to ITP

- **Course Code**: IT2080
- **Course Name**: Information Technology Project (ITP)
- **Credit Points**: 4
- **Duration**: One Semester
- **Enrollment Key** : IT2080

1. Introduction to ITP
Pre-requisites

- Knowledge acquired through one and half years of completing the Associate Diploma Course in Information Technology

1. Introduction to ITP
Mode of Delivery

Formal

- 2 hr Lecture session, 1 hr Tutorial, 2 hr Lab session per week
- Participate for
  - Discussions with the lecturer, Supervisor
  - Activities
  - Presentations, Demonstrations, Viva

1. Introduction to ITP
Mode of Delivery – Lectures

- Lectures are activity based.
- Participation for lectures is compulsory and there are marks allocated for activity participation.

Evaluation (2 hours)

- Evaluation sessions will be conduct only for project evaluations and students are encouraged to spend that time for group meetings, client meetings, activities and project development

1. Introduction to ITP
The staff

Mrs Geethanjali Wimalarathne
Senior Lecturer
Lecturer in charge

Ms Kushnara Suriyawansa
Lecturer

Mr Buddhika Harshanath
Lecturer

Mr Nalaka R. Dissanayake
Senior Lecturer

Mr. Ravi Supunya Swarnakantha
Lecturer
(Matara Center)

Ms Gihani Gunarathna
Lecturer
(Kandy Center)

SLIIT
Discover Your Future

1. Introduction to ITP
Mode of Delivery

Informal

- Group discussions among the group members
- Members participating in project planning and development
- Meeting your client (by appointment)

1. Introduction to ITP
Mode of Delivery – Self-Study

- You should self-study following and use them in your project.
  - Trello (or any other Project Management Tool)
  - Version Control System (VCS) like GIT
  - Juni (Unit Testing)
  - Coding standards
  - SonarQube for code quality
- The reports/ screenshots should be included in the final report as a proof of using those for the project (marks will be allocated)

1. Introduction to ITP Resources

- **Resources and Materials**
  - Unit resources and material will be available in courseweb.
  - Every student must obtain access to the courseweb and enroll to the ITP unit.
  - Refer extra material to gain knowledge and skills
  - Attend sessions organized by SLIIT or any other
- **Notices**
  - All notices will be published on the courseweb unit page
  - Check the unit page regularly

1. Introduction to ITP
Objectives of ITP

What is ITP?

- An important piece of work in your degree program
- Involves students in developing, managing and achieving the objectives of an *ICT project*
- A major independent project activity which involves the **defining of the project objectives**, the preparation of a project documents and showing the various stages of completion of project and develop the product.

1. Introduction to ITP
Objectives of ITP

Why do you need ITP?

- Identify a real-world problem
- Formulate an IT solution to the problem
- Provides an opportunity to demonstrate your knowledge & skills in a practical ICT application
- A benchmark of your ability to solve an industry problem using ICT
- Learn to deal with clients
- Improve teamwork skills
- Opportunity to follow best practices
- Become outstanding graduate

1. Introduction to ITP
Semester plan

Refer to the

2022 S2 IT2080 ITP - Delivery Schedule

in courseweb

1. Introduction to ITP Assessments

- Project Proposal [20%]
  - Project Proposal (document) [10%]
  - Presentation + viva [10%]
- Progress [35%]
  - 80% completed prototype demonstration [25%]
  - Presentation + viva [10%]
- Final product [45%]
  - Report [15%]
  - Product demonstration [20%]
  - Presentation + viva [10%]

1. Introduction to ITP Document Submissions

- Final Charter (revised project charter) + Project proposal – at the end of proposal presentation.
- Group activity documents – at the end of the session
- Final Report – at the end of the final presentation.
  - All the members in the group must contribute
- Group documents can be submitted by one member of the team to the link related to your batch. The document name should contain the groupID. More instructions will be given.

1. Introduction to ITP
Evaluation Criteria

- As specified in the assignments
  - Read and follow all the instructions
  - Refer the marking scheme
  - Discuss with your lecturer and evaluators

1. Introduction to ITP Evaluation

- In every assessment the contribution of each member is assessed
  - **A “common mark” is not given for the entire project**
- To pass the module, every member’s involvement in all project stages and activities are necessary
- Project evaluations are compulsory. Therefore participation is necessary

1. Introduction to ITP Evaluation

- During evaluations, all members must be present
- Absent members will not get any marks for the evaluation
- Evaluations will be held **during the lecture, tutorial or lab sessions** (schedule will be given)
    - Most evaluations are group activities
- Attending all lectures and the tutorial sessions are important
- All the members of the group must attend the same lecture and the tutorial session

2. Project Initiation

2. Project Initiation
Steps to start the Project

1. Form a group (8 Members)
2. Find a project(a client based project)
3. Submit draft charter document (already done)
4. Discuss the project with the ITP lecturers assigned to your batch
5. Present at Proposal Presentation
6. Submit the approved project charter (Finalized charter) and the proposal

SLIIT
Discover Your Future

2. Project Initiation
Categories of Projects

- **Industry Based Projects**
  - These projects must have a real client.
- **Projects aligned with competitions**
  - Example : Microsoft Imagine Cup

2. Project Initiation
Categories of Projects

- **You can develop a**
  - Desktop Application – Not widely used and not recommended (unless there is a special requirement).
  - Web-based Application
    - Not a Website
    - May use browser-based apps and/or mobile apps at client's end

2. Project Initiation
Categories of Projects

Common examples

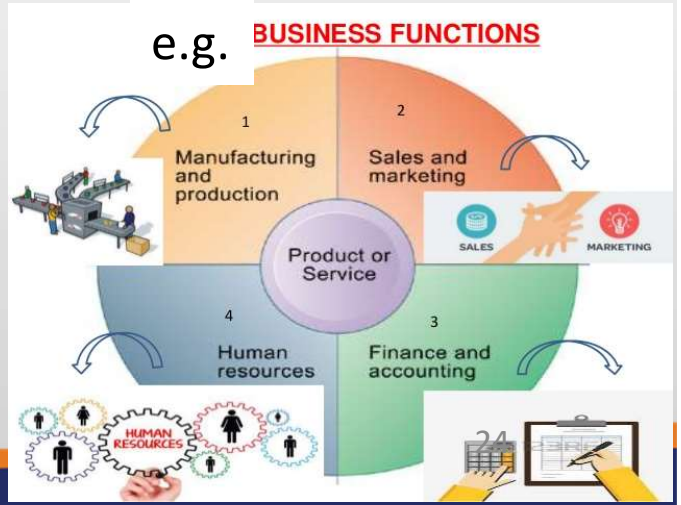
- Vehicle Management System
- Hotel Management System
- Student Management System
- Hospital Management System
- Designing systems (Fashion/Fabric)

2. Project Initiation
Project Scope

- **Select a project with the correct scope (size).**

  - The scope must not be too small or too large for a 12 week duration.
  - The project must have at least 8 significant business functions (1 function per member).
  - Need to have a integrated final system at the end.

e.g.



SLIIT
Discover Your Future

2. Project Initiation
Grouping

- A group with 8 members from same batch.
- A group leader should appoint by the group itself.
- The project group should register using course web respective batch link.

  Note: Ignore this if you have already done.

- Students without a group should talk to respective lecture assigned for the batch to find a group.

2. Project Initiation
Scope for Individual Member

- Every member should be aware of business process
- All the members should have a good overall understanding about the entire project in depth and the functions assigned.
- Develop the functionalities (user stories)
  - meaningful operations
  - Appropriate validations
- Combine(integrate) with others' functions
- Each member should generate at least one meaningful report for the respective function.
- Other than individual function development, all the members should involve in document preparation.

2. Project Initiation
Contribution of the Team Members

- **You must handle your function from "end-to-end".**
  - You are in-charge of documenting, designing, coding, testing, integrating, etc. your function.
  - Maintain a repository, Handle version control (use GitHub)
  - It should not be the case that one member is only writing documents, the other one just doing the ER and the database, another one only designing the user interfaces etc..

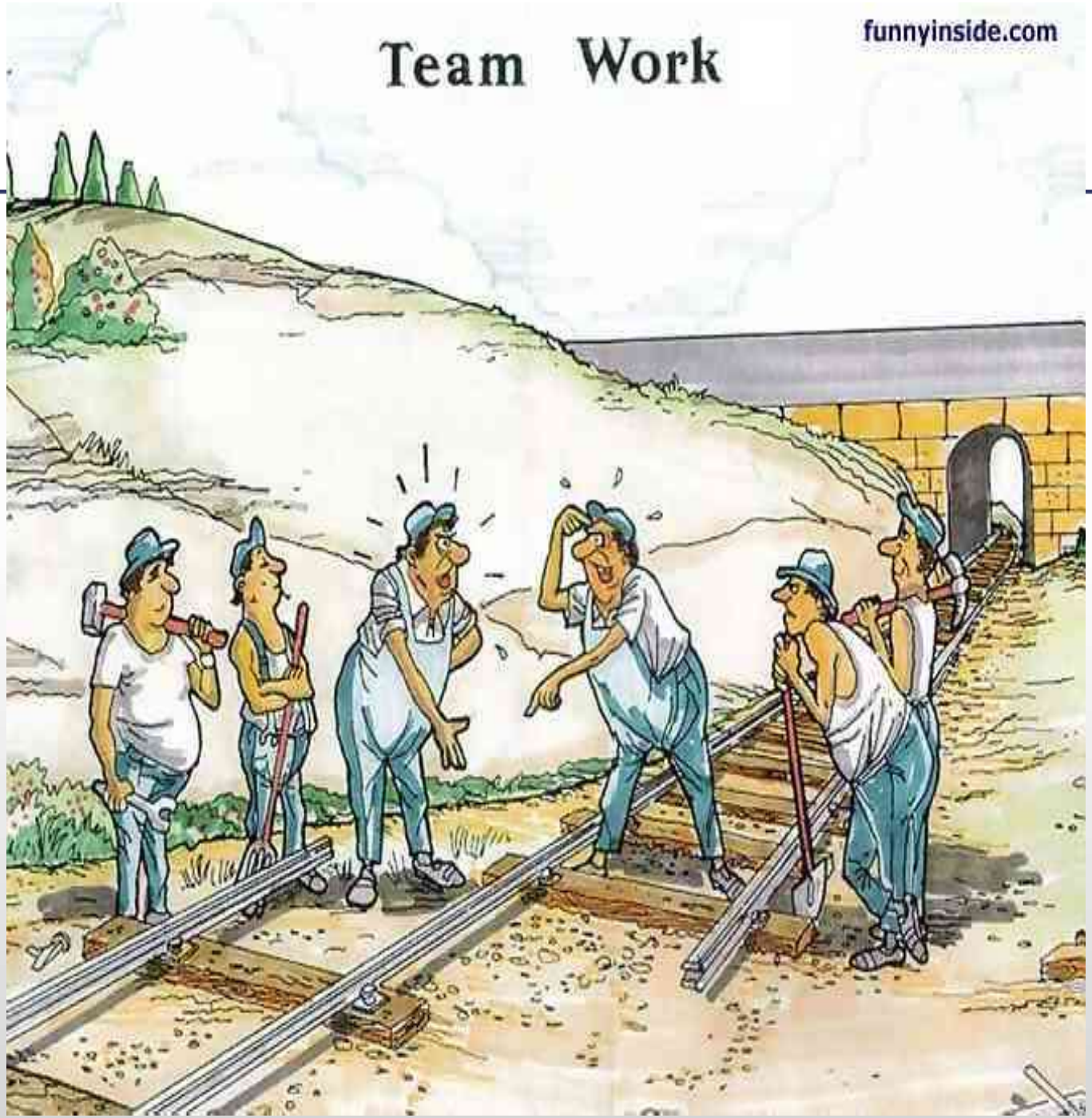
3. Guidelines to Succeed the Project

Why Do Projects Fail?

- People begin programming before they understand the problem
  - A team that begins programming too soon will end up writing good software that solves the wrong problem
- The team has an unrealistic idea about how much work is involved.
  - From far away, most complex problems seem simple to solve
  - Teams can commit to impossible deadlines by being overly optimistic and not thinking through the work

Why Do Projects Fail?

- Defects are injected early but discovered late.
  - Everyone assumes that the testers will catch all of the defects that were injected throughout the project
- The team does not have a good sense of the over all state of the project.
- Lack of team work.



SLIIT
Discover Your Future

How can we make sure that our projects succeed?

- Make sure all decisions are based on **openly shared** information
- All project documents, schedules, estimates, plans and other work products should be shared with the entire team, stakeholders, users and anyone else in the organization who wants them.
- Major decisions that are made about the project should be well-supported and explained.

How can we make sure that our projects succeed?

- Just because a leader/manager has responsibility for a project's success, it doesn't mean that he's more qualified to make decisions than the team members
- Introduce software quality from the very beginning of the project
- Use good engineering practices (coding standards, CICD)
- Managers and teams often want to cut important tasks – especially estimation, reviews, requirements gathering and testing.

Academic Integrity Policy

- Are you aware that following are not accepted in SLIIT.
  - Plagiarism - using work and ideas of other individuals intentionally or unintentionally
  - Collusion - preparing individual assignments together and submitting similar work for assessment.
  - Cheating - obtaining or giving assistance during the course of an examination or assessment without approval
  - Falsification – providing fabricated information or making use of such materials
- Committing above offenses come with serious consequences
- See General support section of Courseweb for full information.

4. Assignment 1 - Project Proposal

Refer the

2022 S2 IT2080 ITP - Assignment 1 – Proposal.pdf

5. Sample Case Studies

5. Sample Case studies
School Management system

- Client: Education Institute/School in your area
- Goal : To automate the school manual work

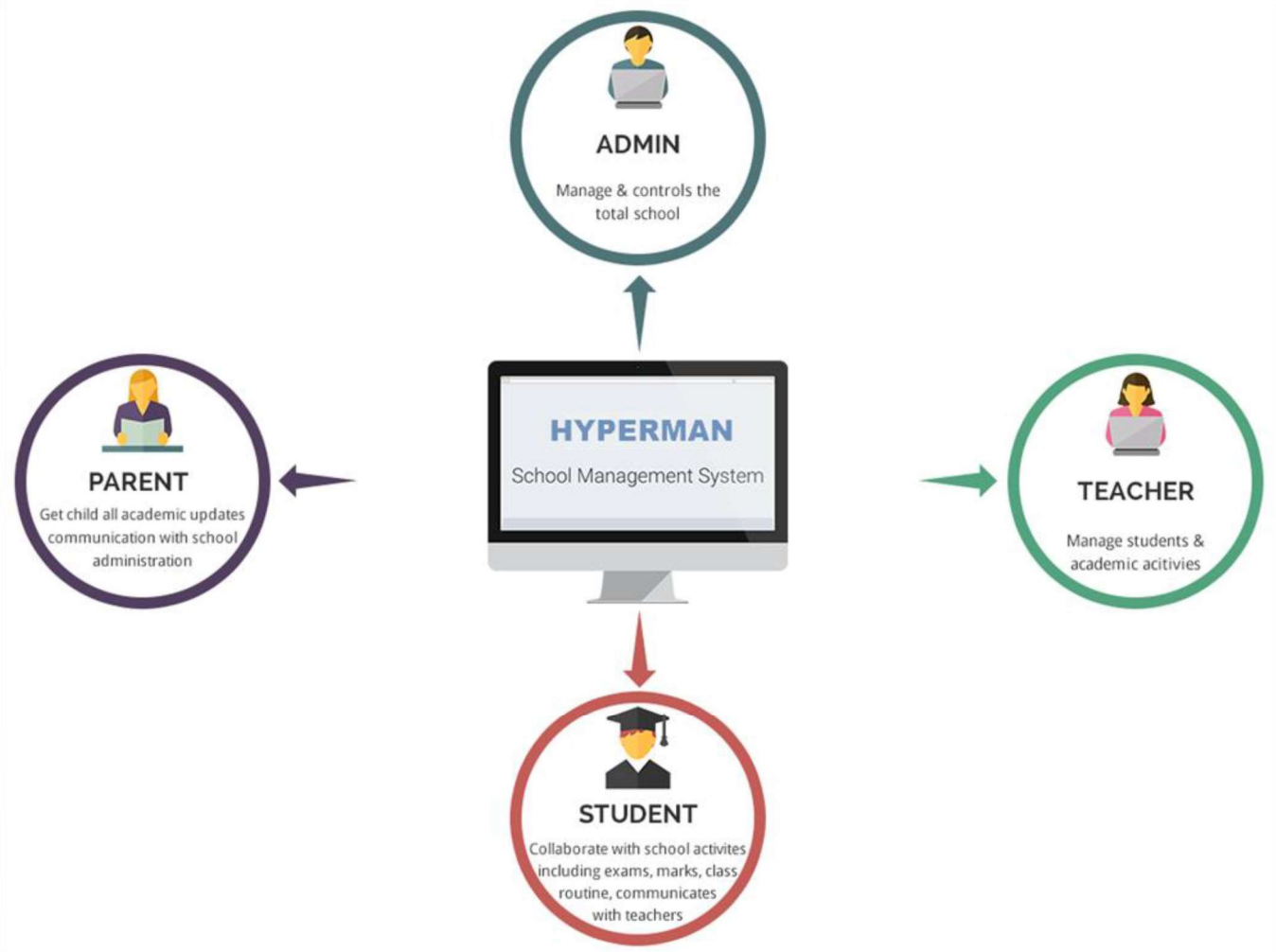
- Identify the main functions of the school
- Identify the people who are involved with the system and why they use the system

5. Sample Case studies
School Management system

School functions can be categorized according to the client requirements

5. Sample Case studies
School Management system

Identify who will going to use the system and what are the benefits from the system.



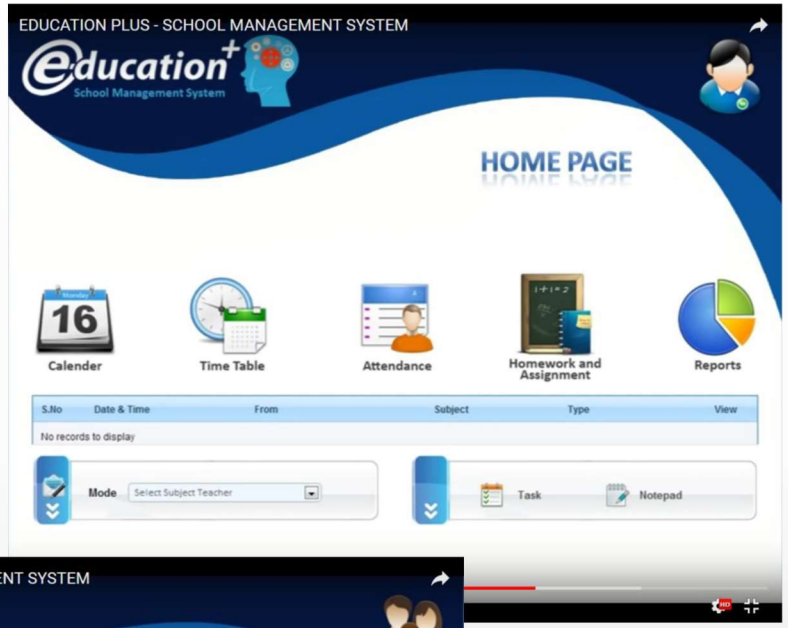
5. Sample Case studies
School Management system

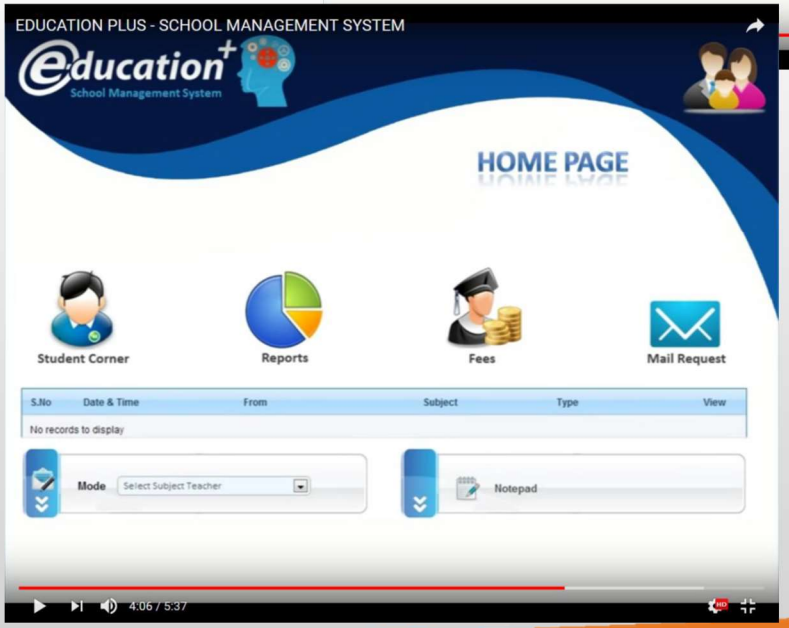
Sample Home page

SLIIT
Discover Your Future

5. Sample Case studies
School Management system

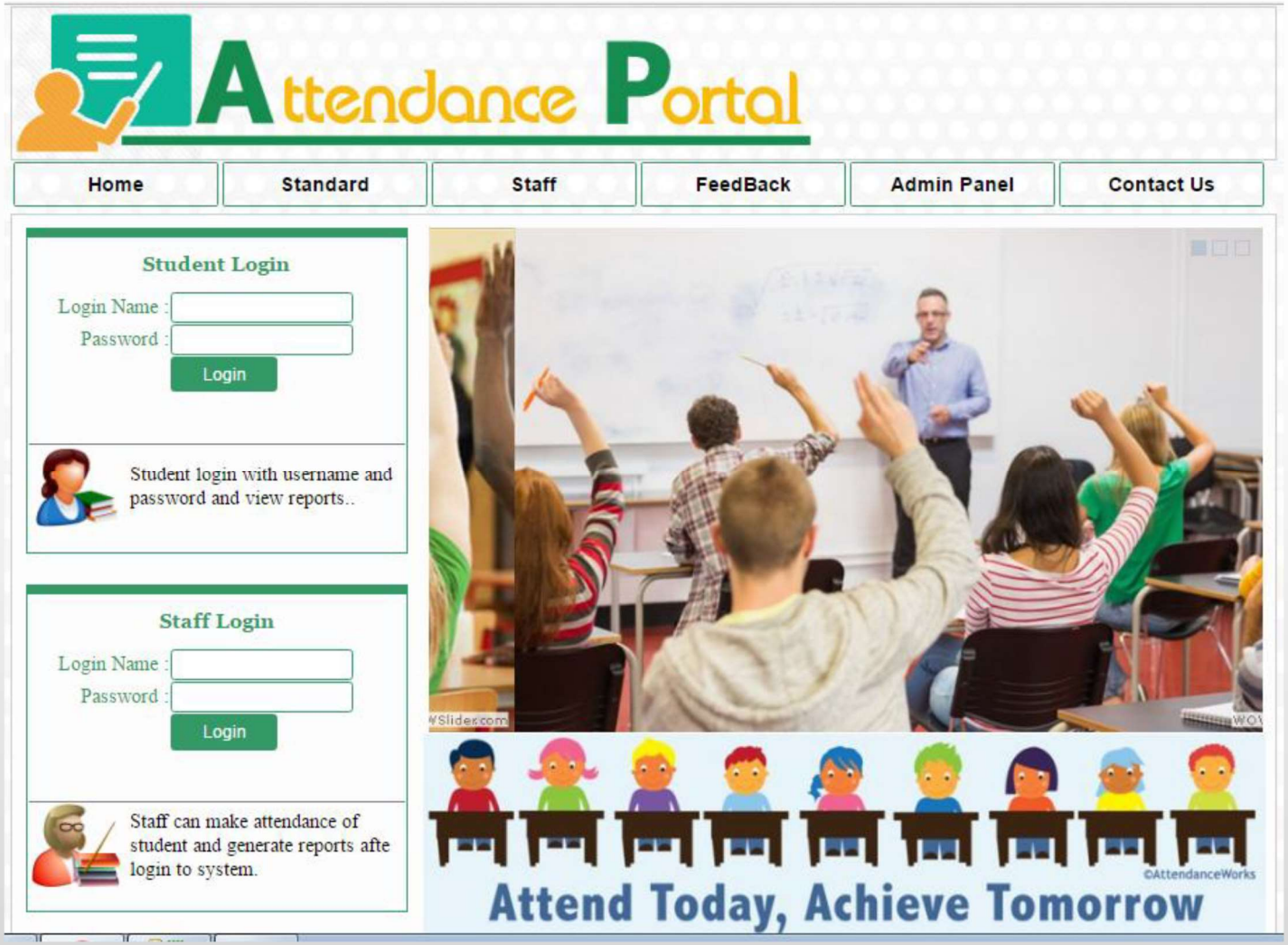
Different user roles may have different home pages





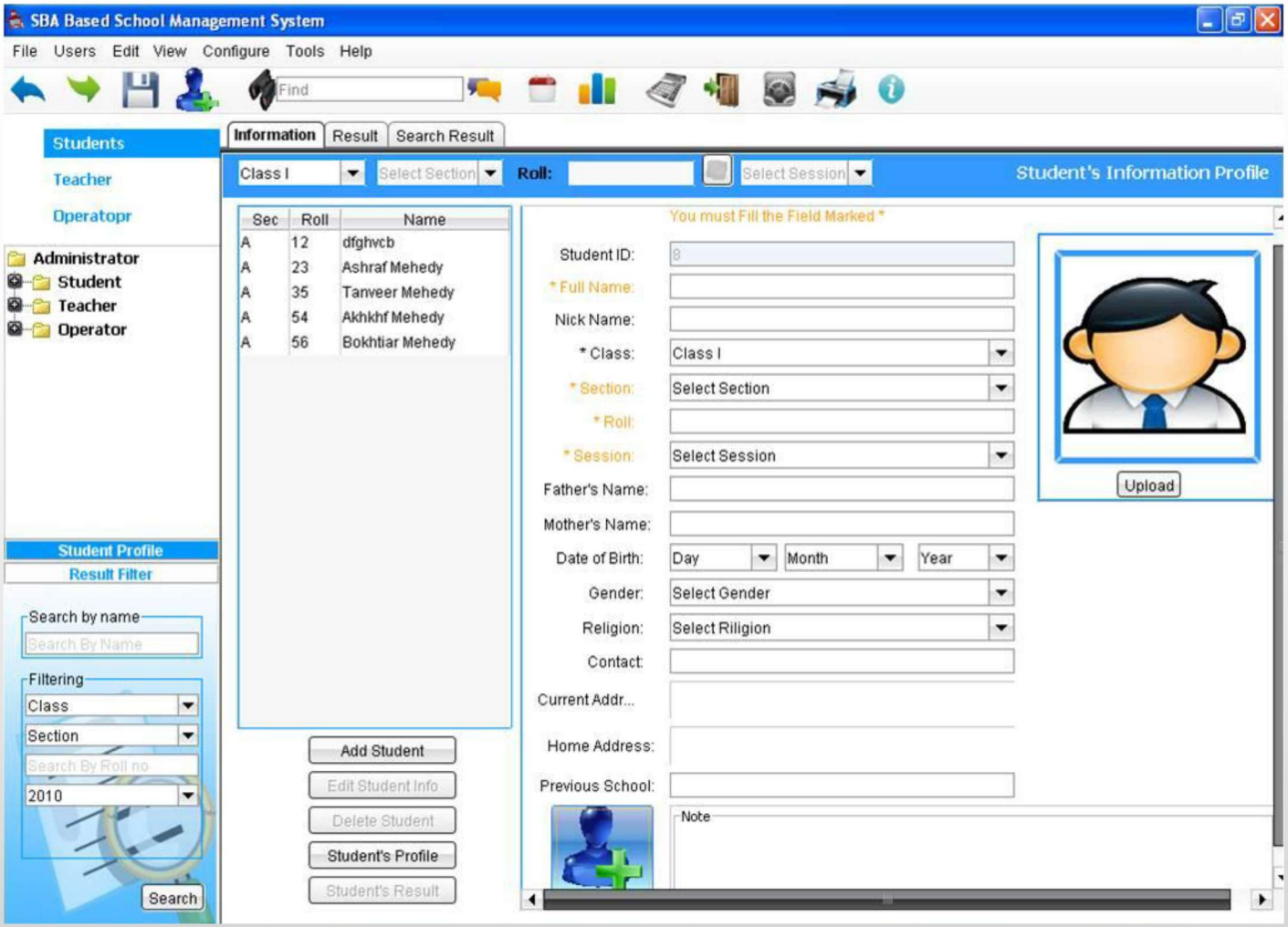
SLIIT
Discover Your Future

5. Sample Case studies
School Management system



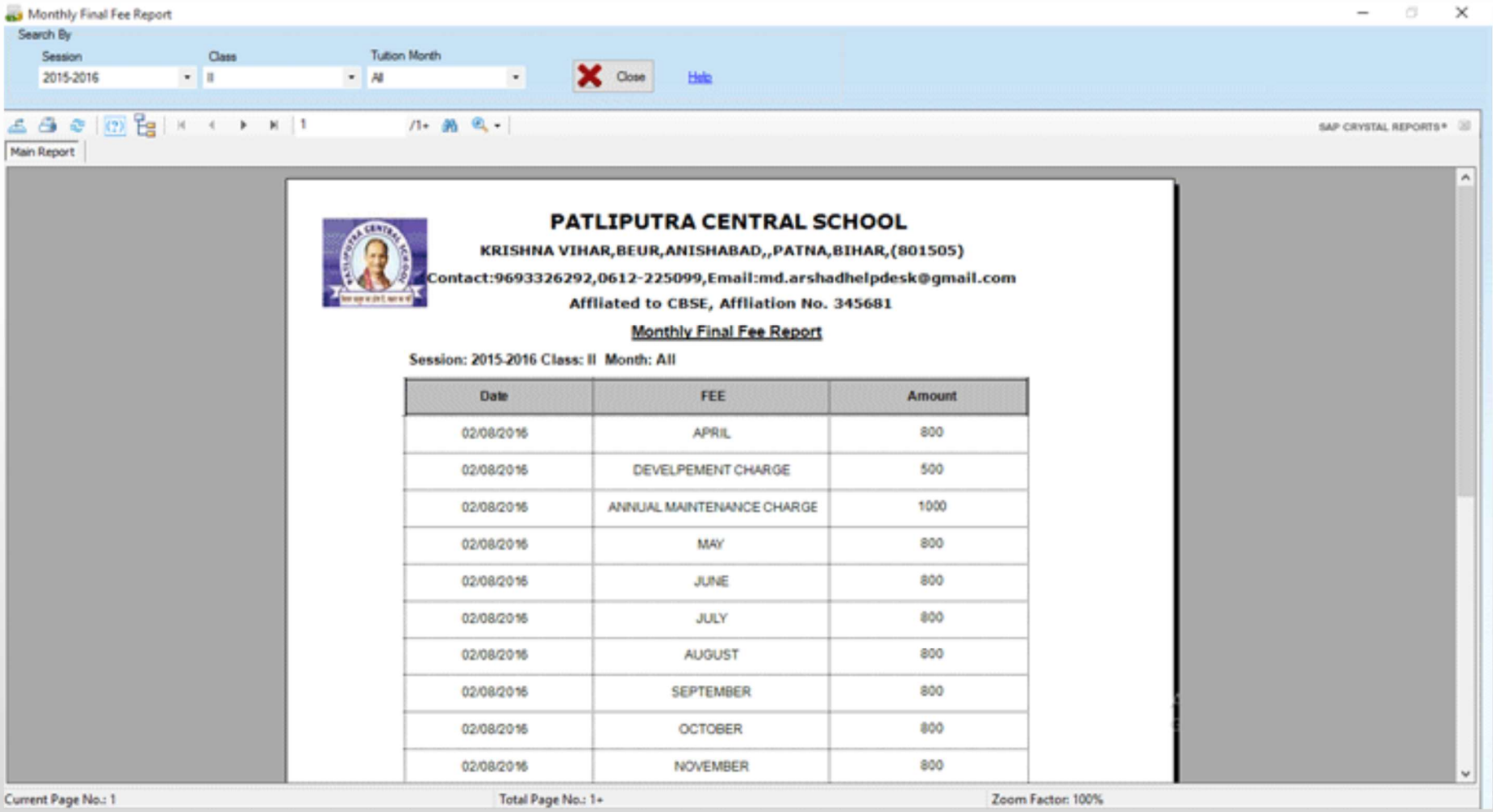
5. Sample Case studies
School Management system

Sample form



5. Sample Case studies
School Management system

Sample report

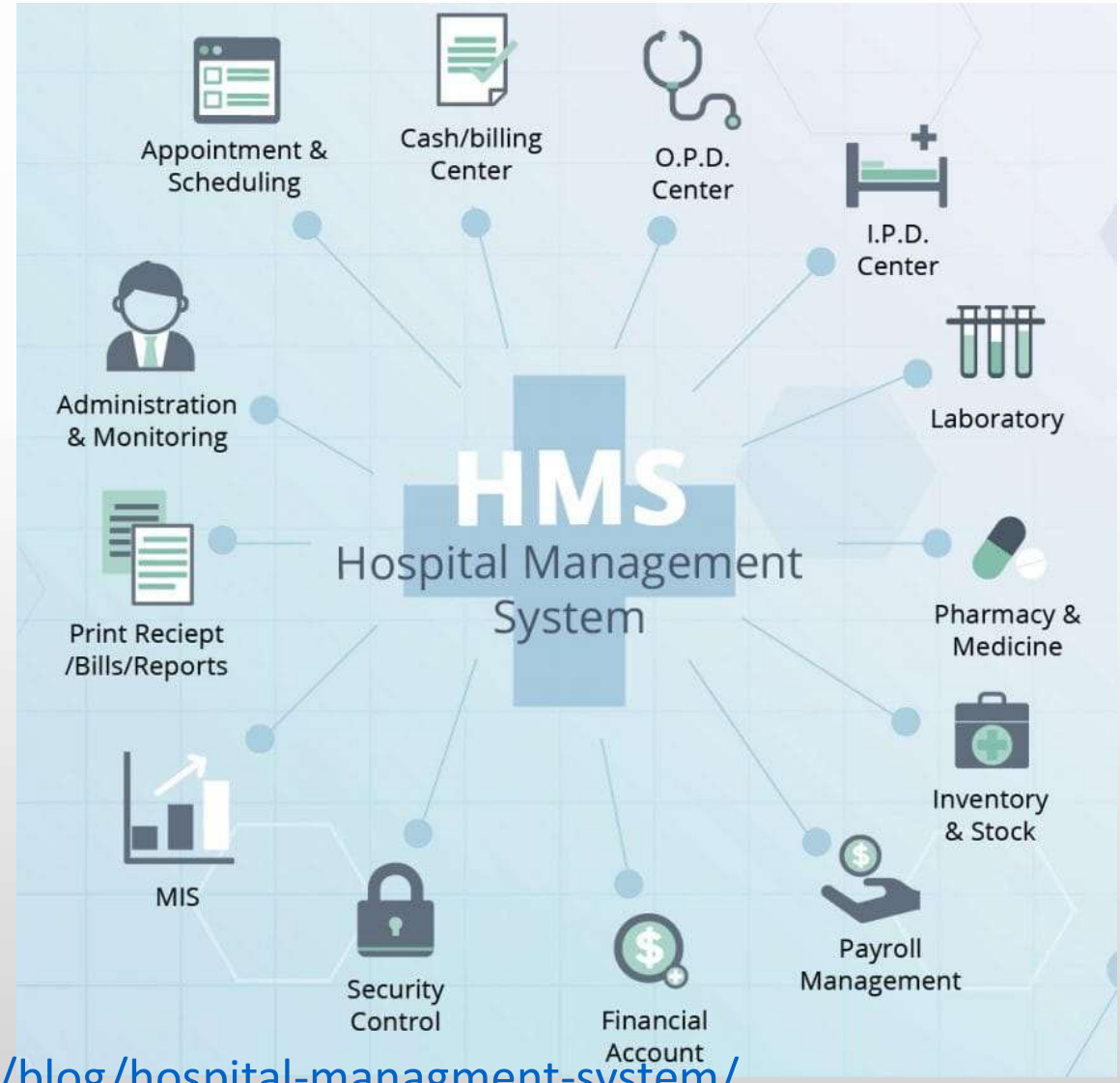


5. Sample Case studies
Hospital Management System

- Client: Hospital in your area.
- Goal : To automate the hospital manual work
- Identify the main functions of the hospital
- Identify the people who are involved with the system and why they use the system

5. Sample Case studies
Hospital Management System

Functions can be categorized according to the client requirements

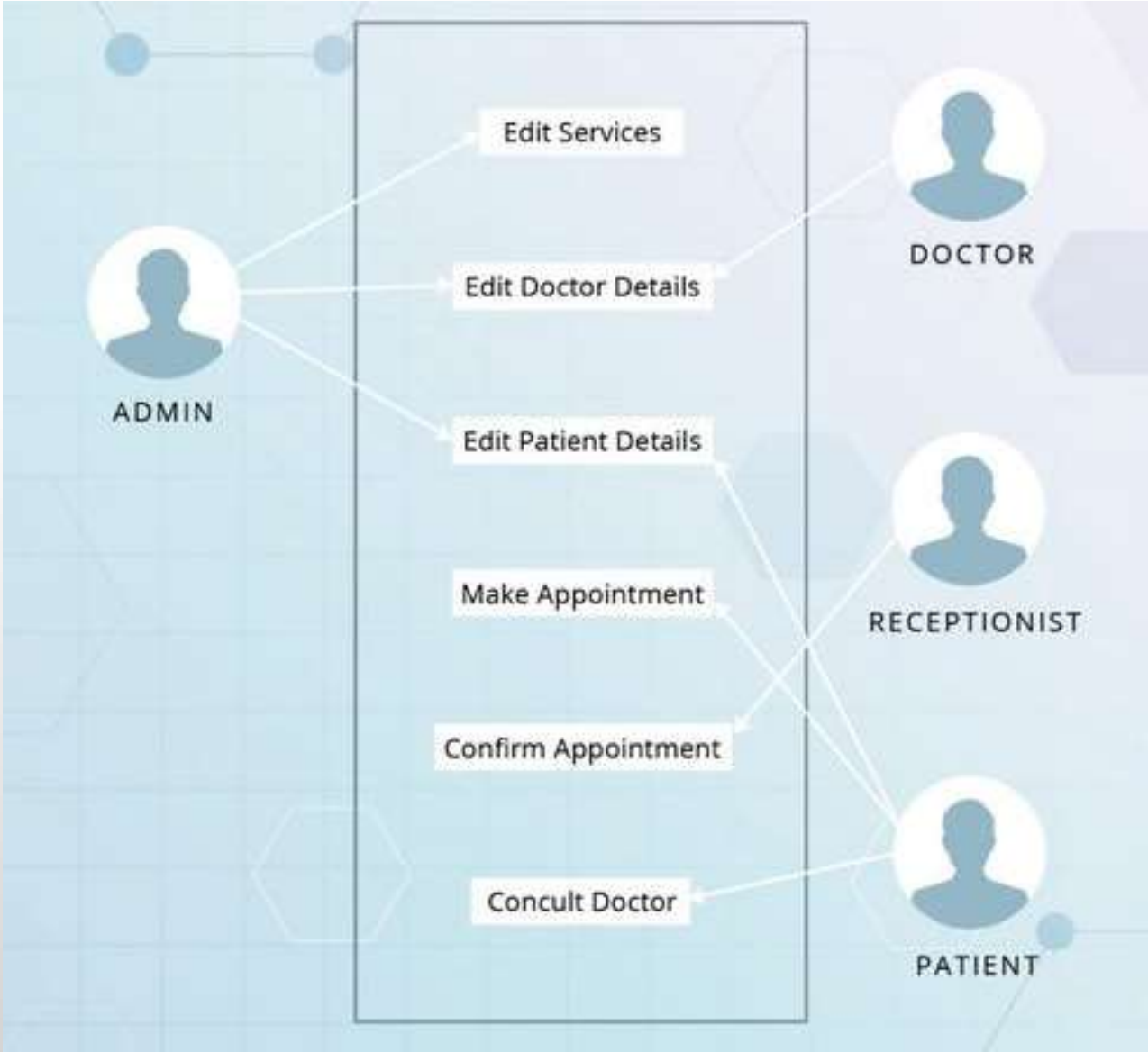


https://existek.com/blog/hospital-managment-system/

5. Sample Case studies
Hospital Management System

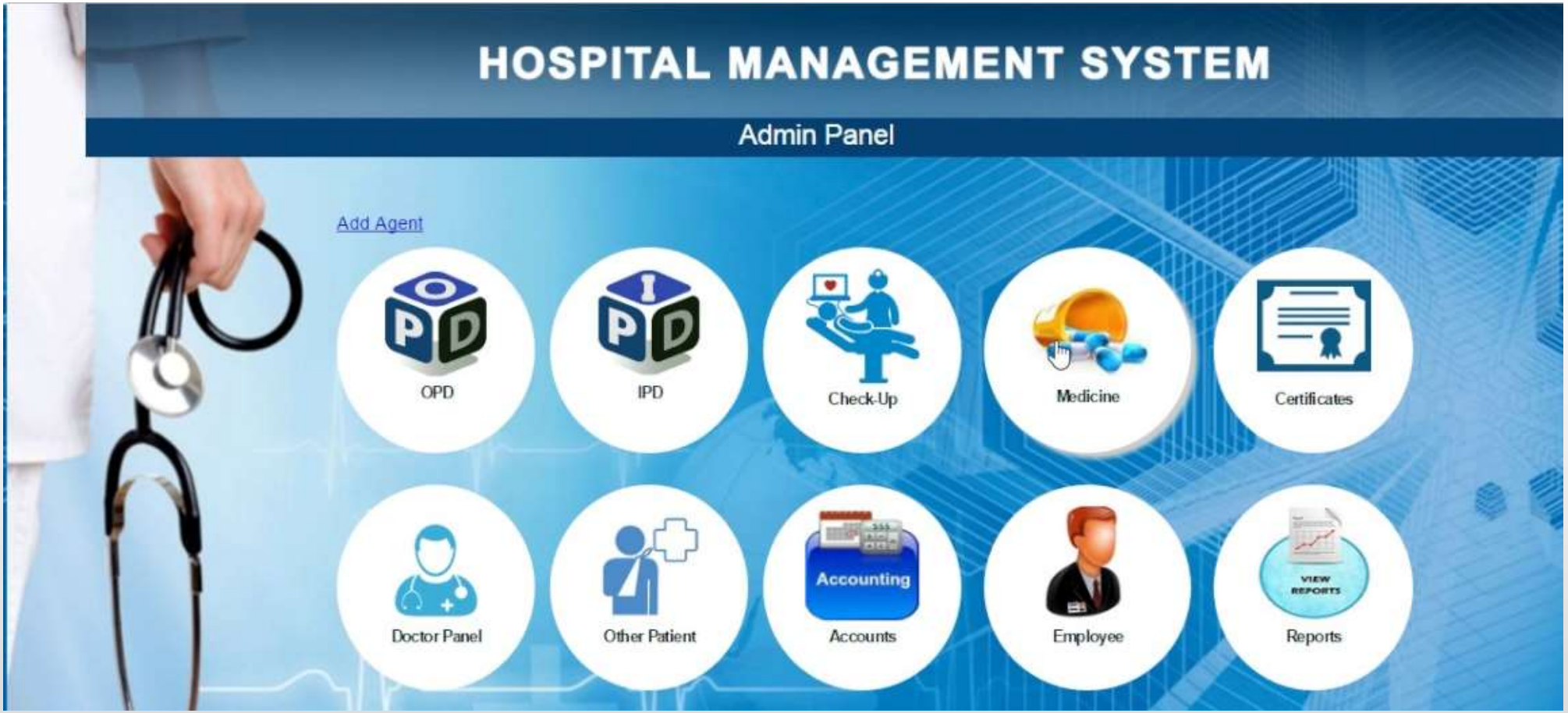
User levels

Identify who will going to use the system and what are the benefits from the system.



5. Sample Case studies
Hospital Management System

Sample home page



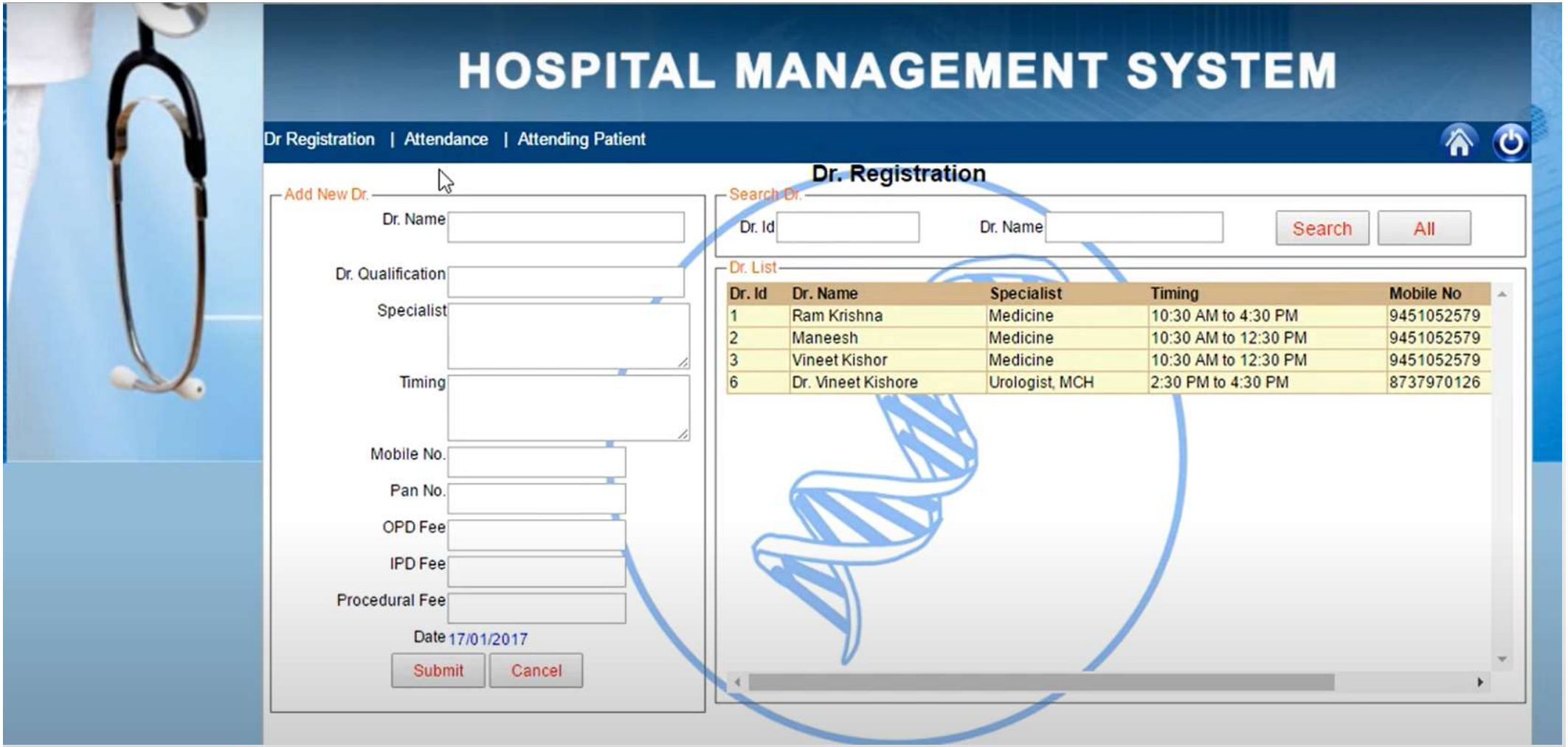
https://www.youtube.com/watch?v=ptaU2NvmqyU

SLIIT
Discover Your Future

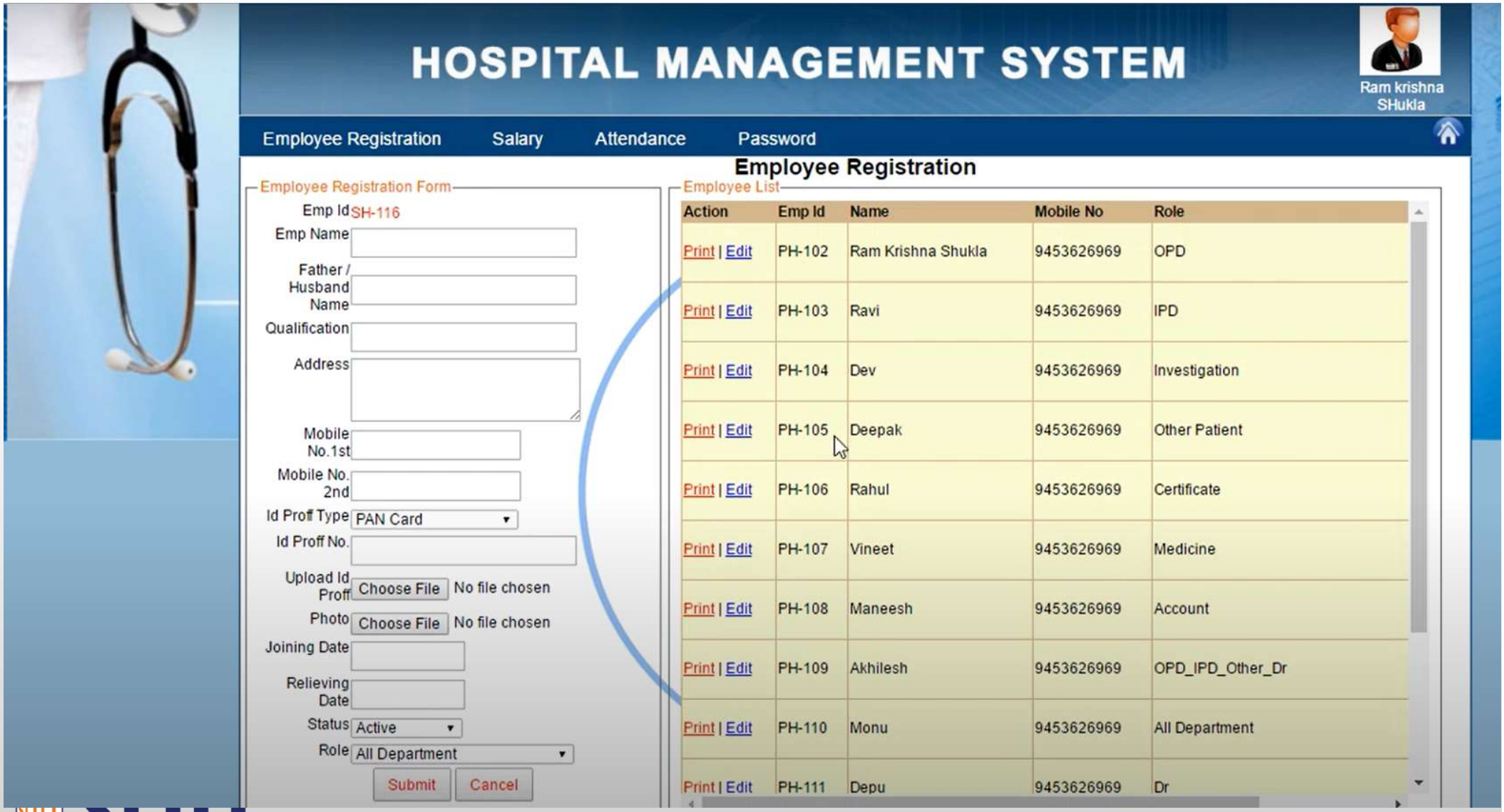
SLIIT
Discover Your Future

5. Sample Case studies
Hospital Management System

5. Sample Case studies
Hospital Management System



5. Sample Case studies
Hospital Management System



Any Questions ???

SLIIT
Discover Your Future

Scope and Schedule Management in Agile

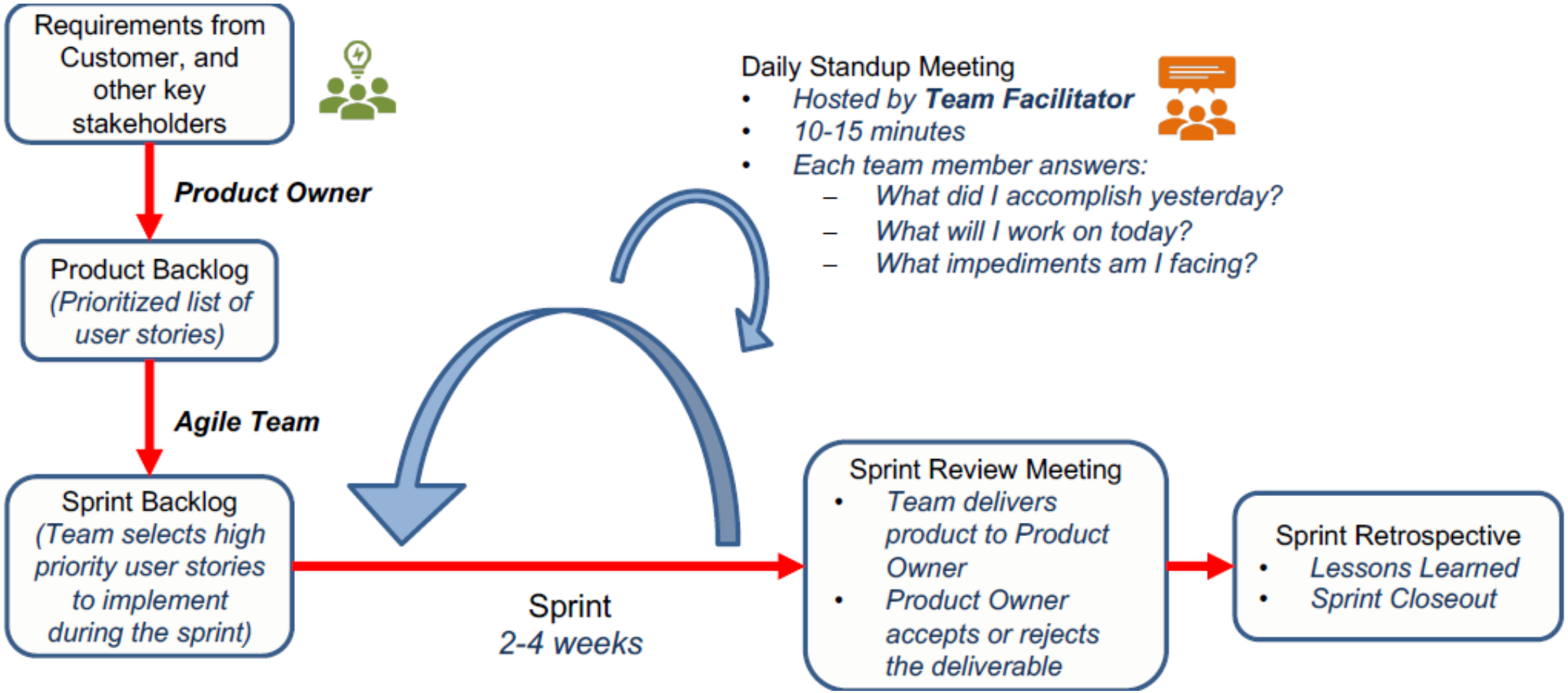
Lecture 2

Sri Lanka Institute of Information Technology
B. Sc. Special Honors in Information Technology
Year 2 – Semester 2

Lifecycle Concepts: The Development Lifecycles

	Predictive	Iterative	Incremental	Agile
Requirements	Fixed	Dynamic	Dynamic	Dynamic
Delivery	Single delivery	Single delivery	Frequent smaller deliveries	Frequent small deliveries (sprints)
Change	Constrained as much as possible	Incorporated at periodic intervals		Incorporated in real-time delivery
Focus	Manage cost	Correctness of solution	Speed	Customer value
Stakeholder involvement	Only at specific intervals or milestones	Regular involvement		Continuously involved
Work	Generally performed once on the project	Repeated until correct	Performed once per increment	Continuously repeated
Best suited for	Well understood projects	Scope determined early but can be modified	Series of iterations that successively add functionality	Rapidly changing environment

Agile Lifecycle



Agile Manifesto

Four Paired Values

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

Project Management Rôles: Agile Rôles

Team Facilitator (Scrum Master)

- *Servant Leader who encourages collaboration between team members*
- *Removes impediments and ensures teams have the tools to complete the work*

Product Owner

- *Represents the business (customer)*
- *Prioritizes user stories to create the product backlog*
- *Approves items delivered during the sprint review*

Project Management Rôles: Agile Rôles

Agile Team

- *Is self organizing and includes developers, testers, business analysts, designers, etc.*
- *Cross functional with T-shaped skills*

Image credit: Palto/Shutterstock

Agile Coach

- *Mentors and coaches the Agile team members*
- *Optional role (often played by experienced team facilitators)*

Project Scope Management in Agile

Agile Concepts for Scope Management

- Product Backlog
  - *Prioritized list of user stories - by the Product Owner*
  - *Sprint backlog/Iteration backlog created from the Product backlog*
  - *Agile team produces high level relative estimates*
  - *Backlog items can be reprioritized at any time (backlog grooming)*
- User Stories

  "As a <Role>, I want <Functionality>, so that <Business benefit>"

  For example:

  "As a call center analyst, I want to search clients by first and last name so I can look up their records quicker."

Agile Concepts for Scope Management

Definition of Done

- Criteria to determine if the work is complete
- Agreed by Product Owner, Agile team and Team Facilitator during Release Planning

Definition of Ready

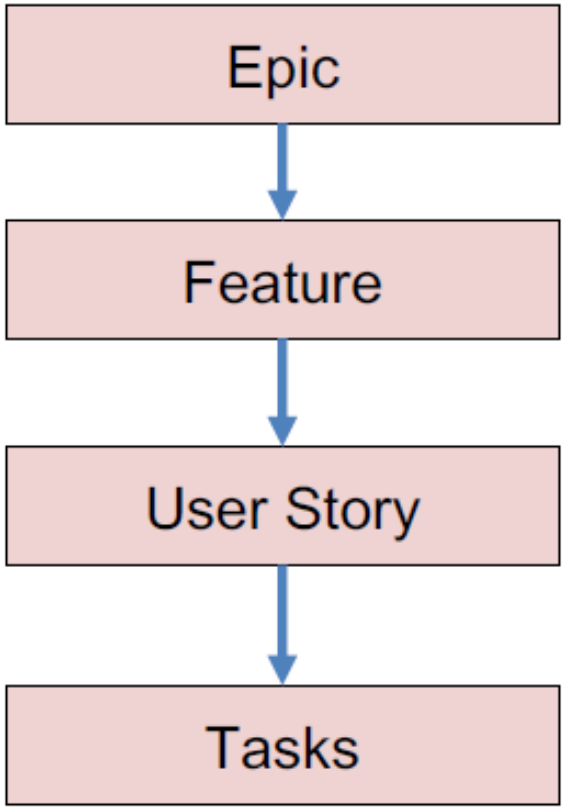
- Checklist to determine that all information is available for the team to start working on the user story
- User stories must be immediately actionable

Project Schedule Management in Agile

Agile Approches to the Schedule

- Agile uses a Product Roadmap (not a schedule)
- Iterative Scheduling with a Product Backlog
  - *User stories are prioritized based on priority and time*
  - *New stories can be added to the backlog*
  - *Does not always work well if there are complex dependency relationships*
- On-demand Scheduling
  - *Team members "pull" work from a queue*
  - *Based on Kanban and Lean methodologies*
  - *Works best when activities can be divided into equal sizes*
  - *Does not always work well if there are complex dependency relationships*

Activities, Relationships and Dependencies in Agile



Very large user story

Capabilities

Functionality

Several tasks needed to fulfill a User story

Agile Estimating Methods, Prioritization and Consensus Gathering

Estimating Methods – Agile

- T-shirt Sizing
  - *Follows sizes of t-shirts such as: XS, S, M, L, XL*
- Story Pointing
  - *Uses the Fibonacci Sequence of numbers*
  - *1, 2, 3, 5, 8, 13, 21, 34, 55*
- Planning Poker
  - *Uses a deck of cards with a modified Fibonacci sequence*

Agile Estimating Methods, Prioritization and Consensus Gathering

Estimating Methods – Agile

- T-shirt Sizing
  - *Follows sizes of t-shirts such as: XS, S, M, L, XL*
- Story Pointing
  - *Uses the Fibonacci Sequence of numbers*
  - *1, 2, 3, 5, 8, 13, 21, 34, 55*
- Planning Poker
  - *Uses a deck of cards with a modified Fibonacci sequence*

Agile Estimating Methods, Prioritization and Consensus Gathering

Prioritization (Agile)

- MoSCoW Analysis
  - *Must Have*
  - *Should Have*
  - *Could Have*
  - *Won't Have*
- Kano Model
  - *Satisfies vs. delights vs. dissatisfies vs. indifferent*
- Paired Comparison Analysis
  - *Prioritization of successive pairs*
- 100 Point Method
  - *Prioritization by spreading points across user stories*

Agile Estimating Methods, Prioritization and Consensus Gathering

Consensus Gathering (Agile)

- Fist of Five
  - *Five fingers = Agree; Fist = Disagree*
  - *1-4 fingers = levels of agreement and disagreement*
- Roman Voting
  - *Thumbs up; Thumbs down*

- Polling
  - *Share a point of view: if unanimous then move on*
  - *If objections raised, then work to solve the objection*
- Dot Voting
  - *Team members use sticky dots to prioritize*

Strategies for Implementing an Agile Development

Kanban:

- Kanban is a popular framework used to implement agile and DevOps software development.
- It requires real-time communication of capacity and full transparency of work.
- Work items are represented visually on a Kanban board, allowing team members to see the state of every piece of work at any time.

Reference: https://youtu.be/iVaFVa7HYj4

Strategies for Implementing an Agile Development

Scrum:

- Scrum is a framework that helps teams work together.
- Scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.

Reference: https://youtu.be/b02ZkndLk1Y

Strategies for Implementing an Agile Development

- "Kanban vs. scrum" is a discussion about two different strategies for implementing an agile development or project management system.

- Kanban methodologies are continuous and more fluid, whereas scrum is based on short, structured work sprints.


THANK YOU
ANY QUESTION?
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor

SLIIT

Discover Your Future

Lecture 03 (Week 05)
Requirements Engineering

IT2080 IT Project

B. Sc. Special Honors in Information Technology

Year 2 – Semester 2

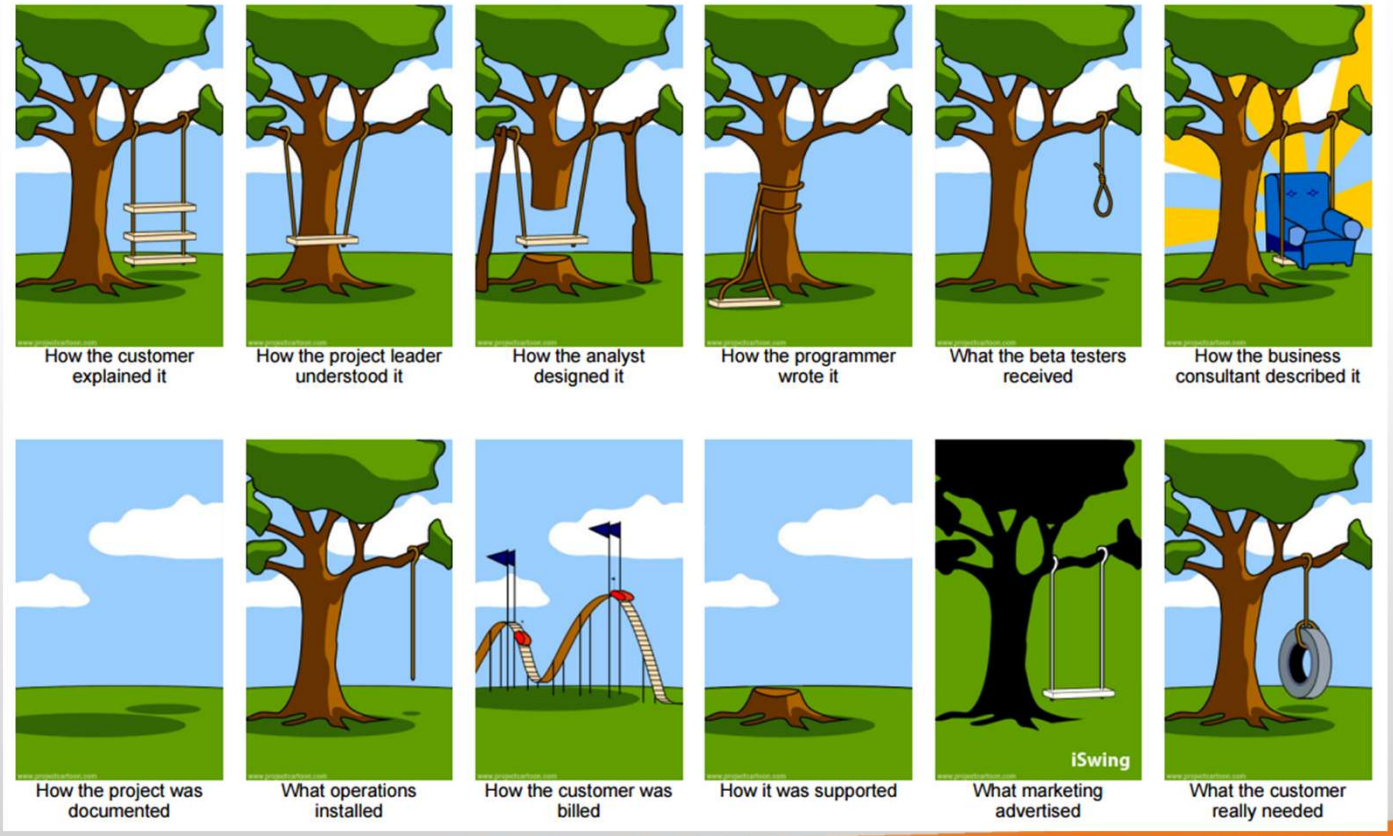
Agenda

1. Requirements Engineering Methods
2. A Case Study

1.
Requirements Engineering Methods

1. Requirements Engineering Methods
Requirements Engineering

Why do we need requirements engineering?



https://i.stack.imgur.com/2YGV6.png

SLIIT
Discover Your Future

1. Requirements Engineering Methods
Requirements Engineering Process

What are the steps in requirements engineering?

1. Requirements Engineering Methods
Requirements Engineering Process

What are the steps in requirements engineering?

1. Requirements elicitation
2. Requirements specification
3. Requirements verification and validation
4. Requirements management

https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/

https://www.javatpoint.com/software-engineering-requirement-engineering

https://www.jamasoftware.com/requirements-management-guide/requirements-gathering-and-management-processes/requirements-engineering

1. Requirements Engineering Methods Requirements Elicitation

First, the requirements should be gathered from different sources.

What are the requirements elicitation methods you have learned?

1. Requirements Engineering Methods Requirements Elicitation

- Background reading
- Documentation reading/analysis
- Surveys (Data surveys and Literature surveys)
- Interviews
- Questionnaires
- User observations
- Workshops
- Brainstorming

https://www.indeed.com/career-advice/career-development/requirement-gathering-techniques

https://www.utm.mx/~caff/doc/OpenUPWeb/openup/guidances/guidelines/req_gathering_techniques_8CB8E44C.html

https://www.tutorialspoint.com/business_analysis/business_analysis_requirement_gathering_techniques.htm

1. Requirements Engineering Methods Requirements Modelling

Secondly, the gathered requirements should be specified/documented.

What are the requirements modelling methods you have learned?

1. Requirements Engineering Methods Requirements Modelling

- Onion diagram – Stakeholder analysis
- User story mapping – FRs, NFRs, TRs
- Use case diagram – FRs, NFRs, TRs
- Wireframes – UI/UX, FRs
- Mind maps – concept and context

https://creately.com/blog/diagrams/requirements-gathering-techniques/

1. Requirements Engineering Methods Requirements Engineering

- When waterfall-based methodologies were used, the requirements were documented and the Software Requirements Specification (SRS) was produced before starting the design and development phases.
- But when Agile-based methodologies are used, heavy documentations are reduced.
- However, it is important to document the requirements for later use.
- When Agile-based methodologies are used, requirements engineering is usually performed in every iteration.

2.
A Case Study

2. A Case Study
Automated Teller Machine (ATM)

Let's assume we have gathered some requirements. The requirements are analyzed and the following user stories are identified.

- Account holder
  - I want to log into the ATM and do these operations
    - Inquire balance
    - Withdraw money
    - Change password
    - Print statement if needed

2. A Case Study
Automated Teller Machine (ATM)

- Cash handler
  - I will place the money in the machine. Then I will log into the ATM system and update the amount of money I placed.

- Branch manager
  - I want to get a status report including the transactions done by the account holders and the ATM's available money .

2. A Case Study
Automated Teller Machine (ATM)

Identify the stakeholders of the ATM system and draw an onion diagram.

- Note that the ATM machine itself is a part of the system.

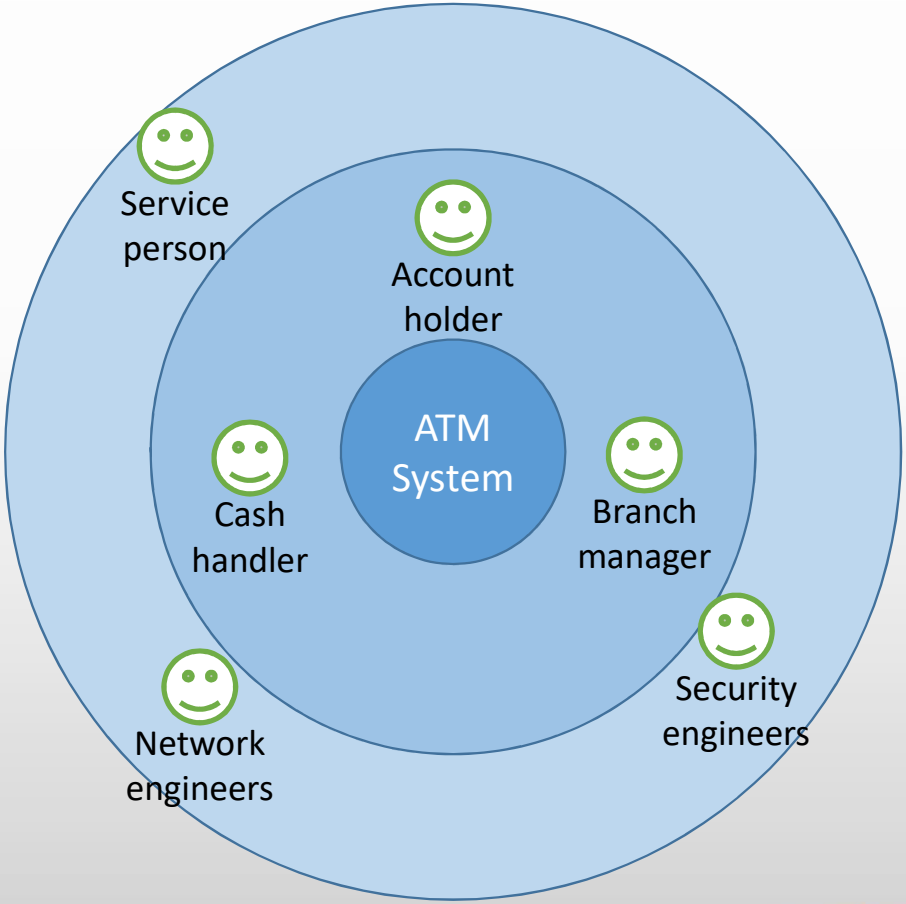
5 mins

2. A Case Study
Automated Teller Machine (ATM)

Stakeholders

Do not include:

- Designer
- Developer
- Tester
- Hosting service
- ……



2. A Case Study
Automated Teller Machine (ATM)

Identify the FRs, NFRs, and TRs from these requirements.

5 mins

2. A Case Study
Automated Teller Machine (ATM)

Functional requirements

- Login – Account holder, Cash handler, Branch manager
- Balance inquiry – Account holder
- Withdraw – Account holder
- Change password – Account holder
- Update cash amount – Cash handler
- Get status report – Branch manager

2. A Case Study
Automated Teller Machine (ATM)

Non-Functional requirements

Security, Scalability, Availability, Usability, Efficiency, Accuracy, Maintainability, ……..

Account holder	**Cash handler**	**Branch manager**
Security	Security	Accuracy
Availability	Accuracy	
Usability		
Performance – Speed		
Accuracy		

https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/
https://www.altexsoft.com/blog/non-functional-requirements/

2. A Case Study
Automated Teller Machine (ATM)

Technical requirements

- Web-based system
- Back-end – Java web service
  - RESTful API
  - Host – server specification
- Front-end – Java Swing desktop app, Linux OS
- Security – SSL
- Database – Oracle

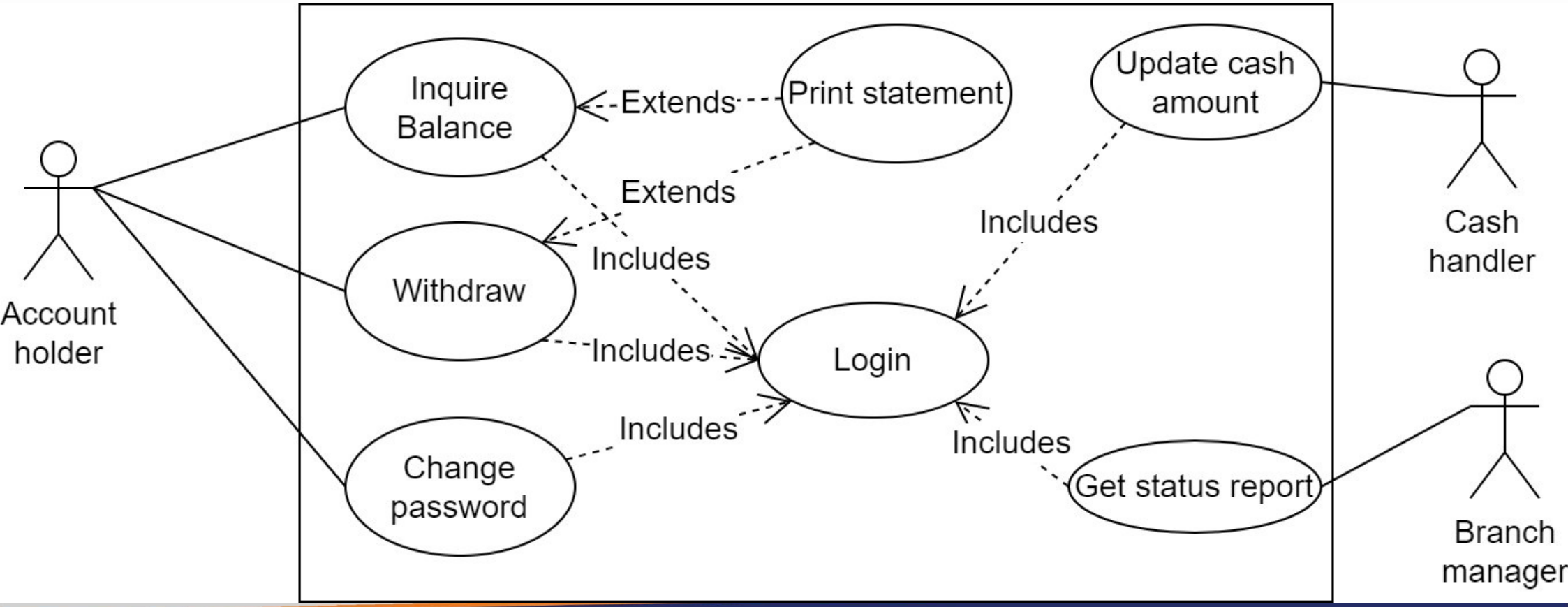
2. A Case Study
Automated Teller Machine (ATM)

Draw a use case diagram for the ATM system

5 mins

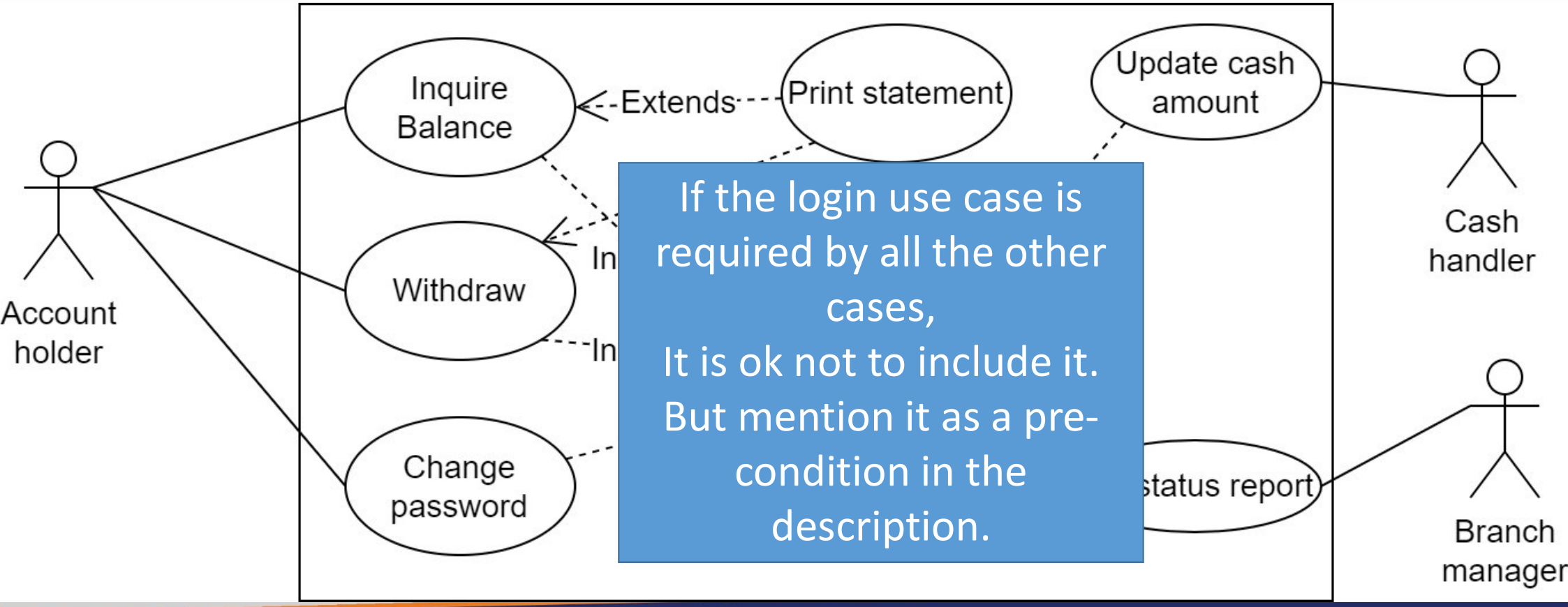
2. A Case Study
Automated Teller Machine (ATM)

Use case diagram



2. A Case Study
Automated Teller Machine (ATM)

Use case diagram



2. A Case Study
Automated Teller Machine (ATM)

Use case description

Use case name:	
Actor:	
Goal:	
Overview:	
Pre-conditions:	
Post-conditions:	
Basic path/ alternative path:	
NFRs and TRs:	

https://www.visual-paradigm.com/guide/use-case/what-is-use-case-specification/

https://businessanalystmentor.com/use-cases-the-use-case-narrative/

Summary

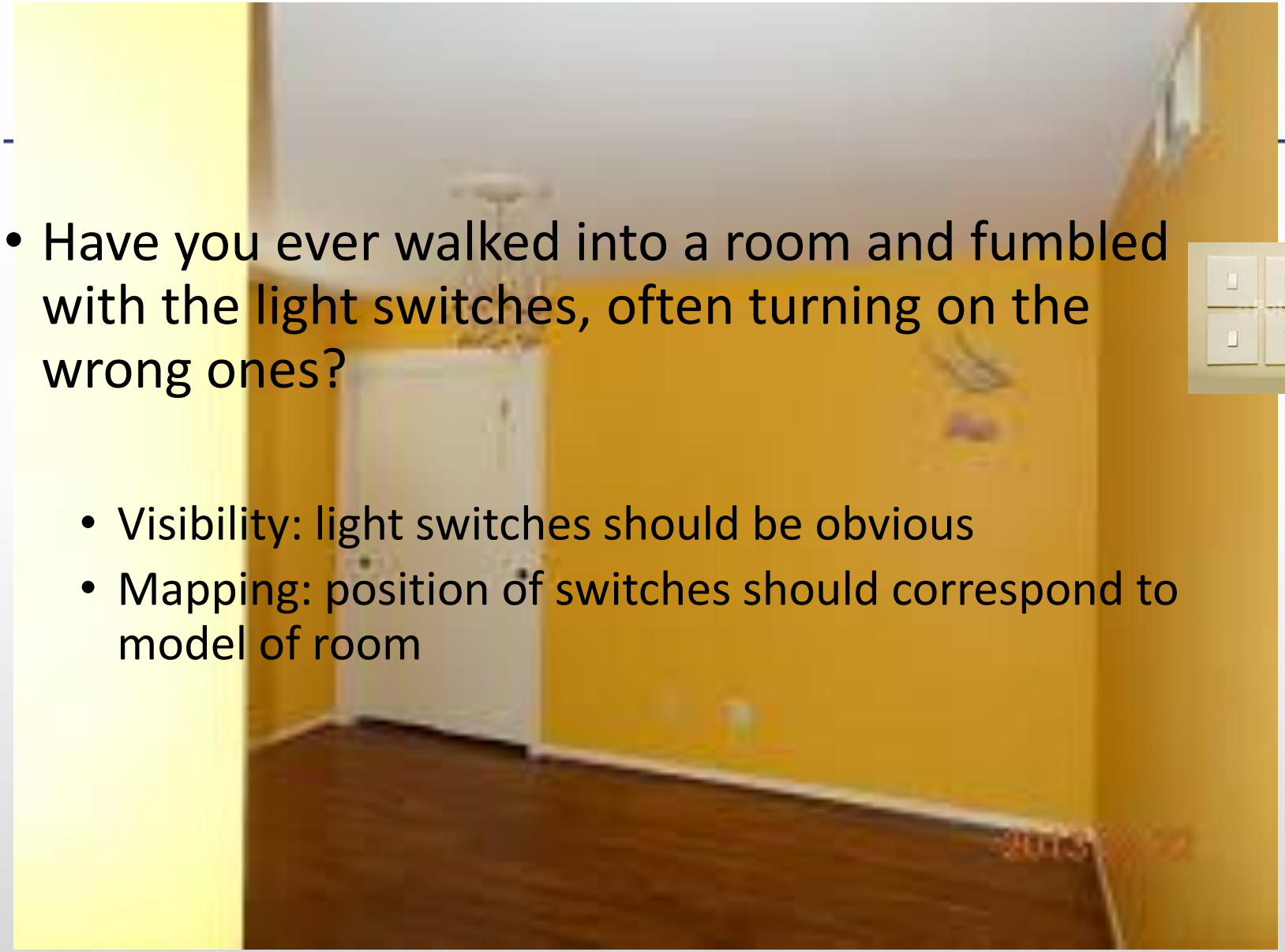
1. Requirements Engineering Methods
2. A Case Study

SLIIT
Discover Your Future

USER INTERFACE DESIGN

Lecture 3

Sri Lanka Institute of Information Technology
B. Sc. Special Honors Degree In Information Technology
Year 2 – Semester 2



- Have you ever walked into a room and fumbled with the light switches, often turning on the wrong ones?

  - Visibility: light switches should be obvious
  - Mapping: position of switches should correspond to model of room

The User Interface

- "The user interface is the most important part of any computer system." (Galitz, 2002, p. 1)

It can be seen,

it can be heard and

It can be touched

"The best interface is the one that is not noticed, one that permits the user to focus on the information and task at hand, not the mechanisms used to present the information and perform the task." (Galitz, 2002, p. 4)

Source: *The Essential Guide to User Interface Design* by Galitz

What Comprises Good Design?

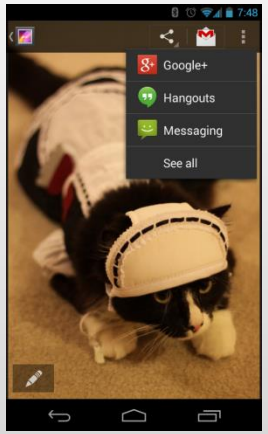
- Understanding of:
  - people, how we see, understand, and think
  - how information must be visually presented to enhance human acceptance and comprehension
  - how eye and hand movements must flow to minimize the potential for fatigue and injury

Benefits of Good Design

- Higher task completion rates
- More efficient task completion rates
- Reduced training costs
- Improved customer service

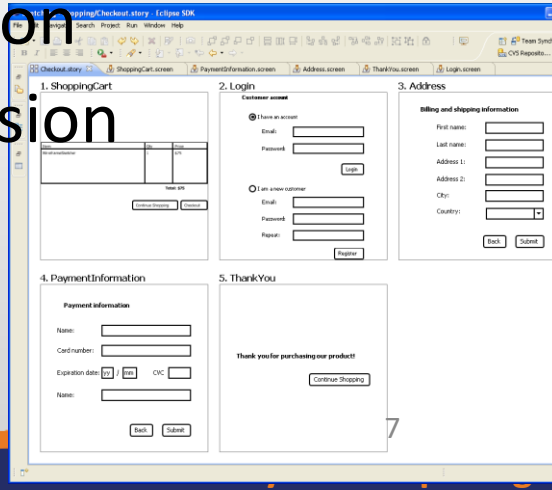
Advantages of GUI

- Symbols recognized faster than text
- Faster learning
- Easier remembering
- More natural
- Exploits visual/spatial cues
- Increased feeling of control
- Immediate feedback
- Predictable system responses
- Low typing requirements
- Fewer errors



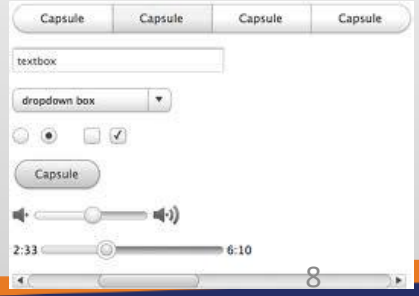
Disadvantages

- Greater design complexity
- Learning still necessary
- Lack of experimentally-derived design guidelines
- Inconsistencies in technique and terminology
- Not always familiar
- Not always the preferred style of interaction
- Not always the fastest style of interaction
- Increased chances of clutter and confusion
- May consume more screen space



Guidelines for the Interface Design

1. Know your user or client. UIs should be designed to match the skills, experience and expectations of its anticipated users.
2. Understand the basic business function
3. Understand the principles of good screen design
4. Develop system menus and navigation schemes
5. Select the proper kind of windows
6. Select the proper device-based controls



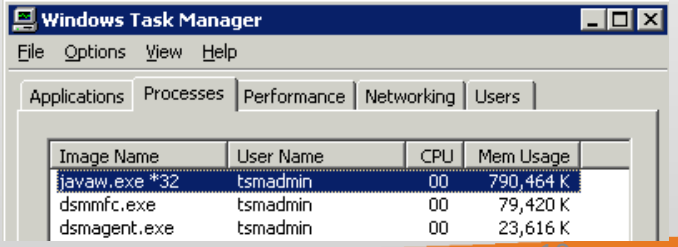
Source: *The Essential Guide to User Interface Design* by Galitz

Guidelines for the Interface Design (cont)

7. Choose the proper screen-based controls
8. Write clear text and messages
9. Provide effective feedback and guidance and assistance
10. Provide effective internationalization and accessibility
12. Create meaningful graphics, icons, and images
13. Choose the proper colors
14. Organize and layout windows and pages

Human Factors In Interface Design

- Limited short-term memory
  - People can instantaneously remember about 7 items of information. If you present more than this, they are more liable to make mistakes.
- People make mistakes
  - When people make mistakes and systems go wrong, inappropriate alarms and messages can increase stress and hence the likelihood of more mistakes.
- People are different
  - People have a wide range of physical capabilities. Designers should not just design for their own capabilities.
- People have different interaction preferences
  - Some like pictures, some like text.



User Interface Design Principles

Principle	Description
User familiarity	The interface should use terms and concepts which are drawn from the experience of the people who will make most use of the system.
Consistency	The interface should be consistent in that, wherever possible, comparable operations should be activated in the same way.
Minimal surprise	Users should never be surprised by the behaviour of a system.
Recoverability	The interface should include mechanisms to allow users to recover from errors.
User guidance	The interface should provide meaningful feedback when errors occur and provide context-sensitive user help facilities.
User diversity	The interface should provide appropriate interaction facilities for different types of system user.

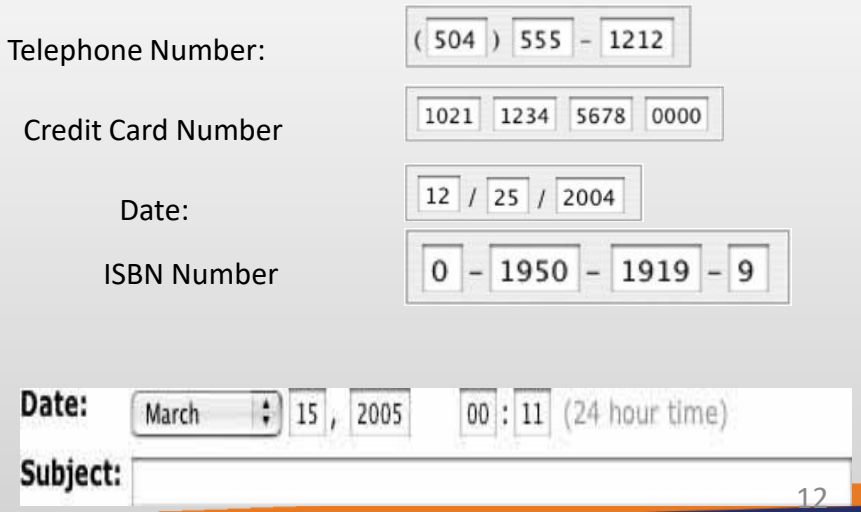
UI Design principles contd.

User familiarity

The interface should be based on user-oriented terms and concepts rather than computer concepts

E.g., an office system should use concepts such as letters, documents, folders etc. rather than directories, file identifiers, etc.

- Ease of learning
- Ease of use
- Ease of remembering



UI Design Principles Contd.

Consistency

- A system should look, act, and operate the same throughout.

  Similar components should:
  - Have a similar look
  - Have similar uses
  - Operate similarly
- Commands and menus should have same
  - Format / appearance
  - Command punctuation
  - Layout
  - Abbreviations / keyboard shortcuts
- Error-messages
  - Same styles / formats from place to place
  - Same location on screen when they appear

UI Design Principles Contd.

Minimal surprise

- If a command operates in a known way, the user should be able to predict the operation of comparable commands

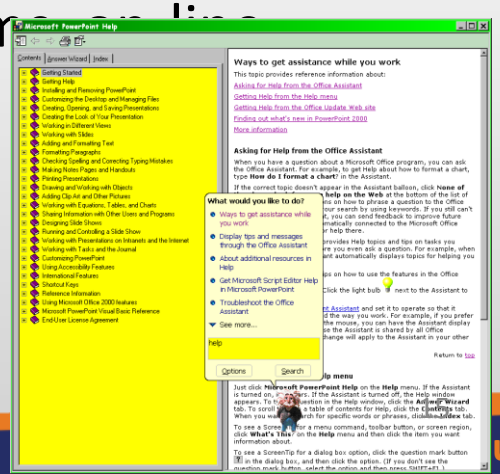
UI Design Principles Contd.

Recoverability

- The system should provide some resilience to user errors and allow the user to recover from errors. This might include an undo facility, confirmation of destructive actions, 'soft' deletes, etc.

User guidance

- Some user guidance such as help system manuals, etc. should be supplied



UI Design Principles Contd.

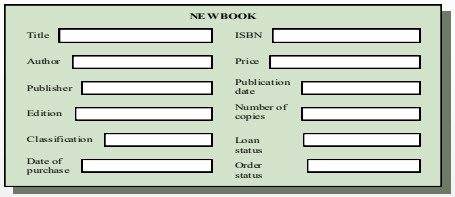
User diversity

- Interaction facilities for different types of user should be supported.

E.g some users have seeing difficulties and so larger text should be available

Interaction Styles

- Direct manipulation
- Menu selection
- Form fill-in
- Command language
- Natural language

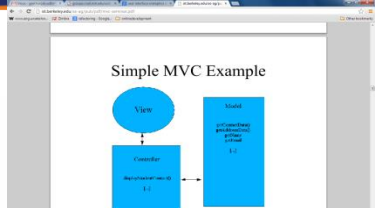


Advantages & Disadvantages

Interaction style	Main advantages	Main disadvantages	Application examples
Direct manipulation	Fast and intuitive interaction Easy to learn	May be hard to implement Only suitable where there is a visual metaphor for tasks and objects	Video games CAD systems
Menu selection	Avoids user error Little typing required	Slow for experienced users Can become complex if many menu options	Most general-purpose systems
Form fill-in	Simple data entry Easy to learn	Takes up a lot of screen space	Stock control, Personal loan processing
Command language	Powerful and flexible	Hard to learn Poor error management	Operating systems, Library information retrieval systems
Natural language	Accessible to casual users Easily extended	Requires more typing Natural language understanding systems are unreliable	Timetable systems WWW information retrieval systems

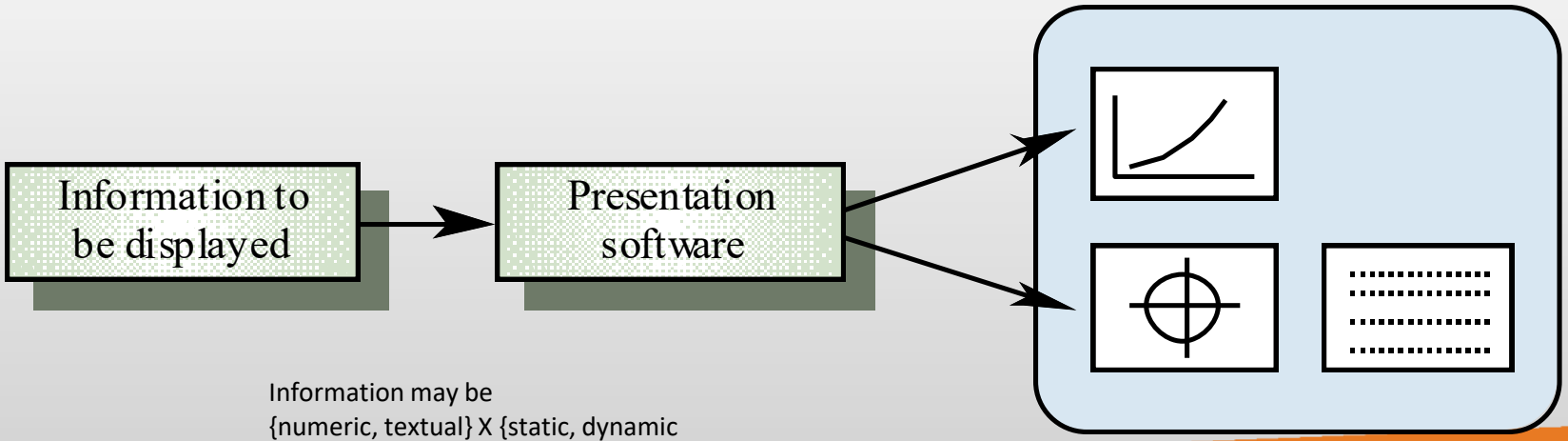
User-system Interaction

- Two problems must be addressed in interactive systems design
  - How should information from the user be provided to the computer system?
  - How should information from the computer system be presented to the user?
- User interaction and information presentation may be integrated through a coherent framework such as a user interface metaphor.



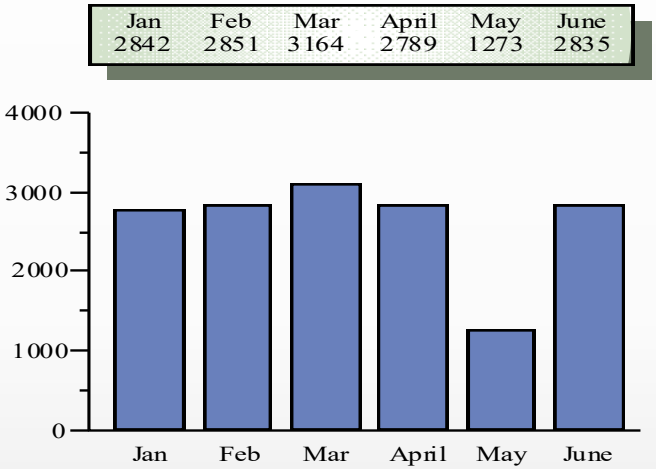
Information Presentation

- Information presentation is concerned with presenting system information to system users
- The information may be presented directly or may be transformed in some way for presentation
- The Model-View-Controller approach is a way of supporting multiple presentations of data



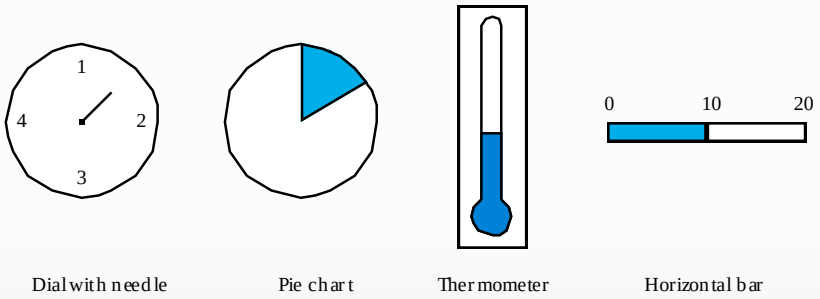
Information may be
{numeric, textual} X {static, dynamic

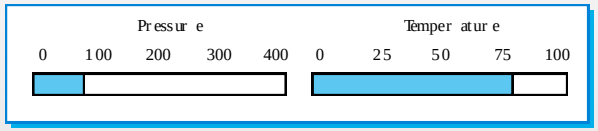
Information Presentation



Digital presentation

- Compact - takes up little screen space;
- Precise values can be communicated



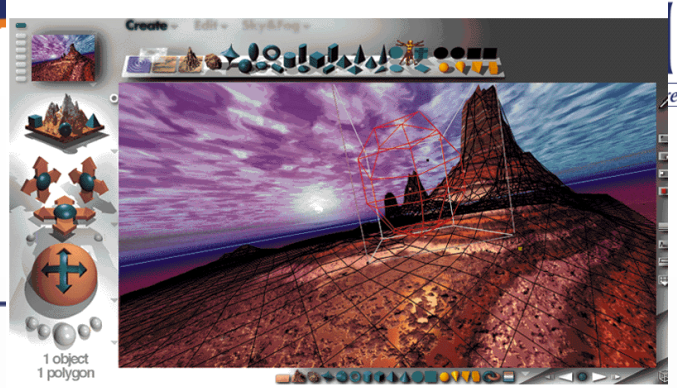


Displaying relative values

Analogue presentation

- Easier to get an 'at a glance' impression of a value;
- Possible to show relative values;
- Easier to see exceptional data values.

Data Visualisation



The GUI Created a Revolution

GUIs allow the most imaginative interfaces to be created; witness this Bryce 3-D modeling program. All the symbols are working tools. For example, the ones on the left side with crosses are camera controls. The large symbol at the bottom is the camera trackball, while the other three control the x, y and z axes. On top are primitive graphic elements, including an "organic rock generator," fourth from left. *(Screen shot courtesy of MetaCreations Corporation.)*

- Concerned with techniques for displaying large amounts of information.
- Possible data visualisations are:
  - Weather information collected from a number of sources;
  - The state of a telephone network as a linked set of nodes;
  - Chemical plant visualised by showing pressures and temperatures in a linked set of tanks and pipes;
  - A model of a molecule displayed in 3 dimensions;
  - Web pages displayed as a hyperbolic tree.

Colours

Colors are not just for decorative purposes; colors sell ideas, products, and, services.

Color Theory actually covers a number of things, but at the most basic level it is the interaction of colors in a design through

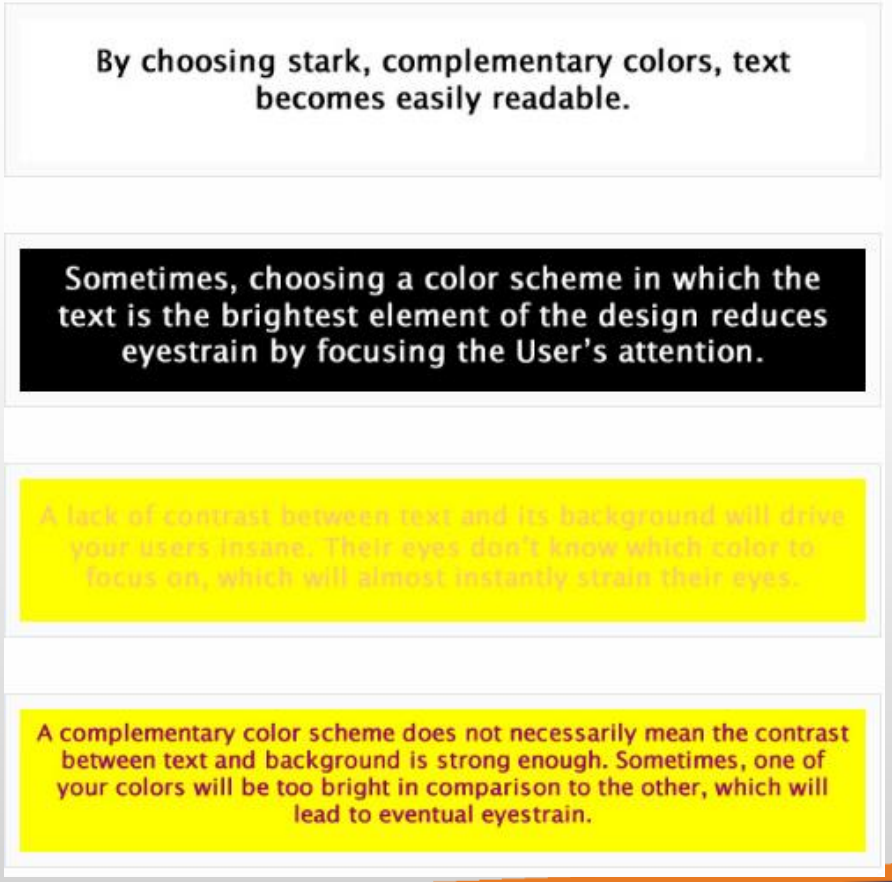
- complementation
- contrast
- vibrancy.

Complementation

Complementation refers to the way we see colors in terms of their relationships with other colors. When colors occupy opposite ends of the color spectrum, they lead people to consider a design visually appealing by establishing a happy medium the eye can reside in. Rather than straining to accommodate for a particular area of the color spectrum, the eye is provided a balance.

Contrast

Contrast reduces eyestrain and focuses user attention by clearly dividing elements on a page. The most apparent example of contrast is an effective selection of background and text color, as shown below:



Contrast

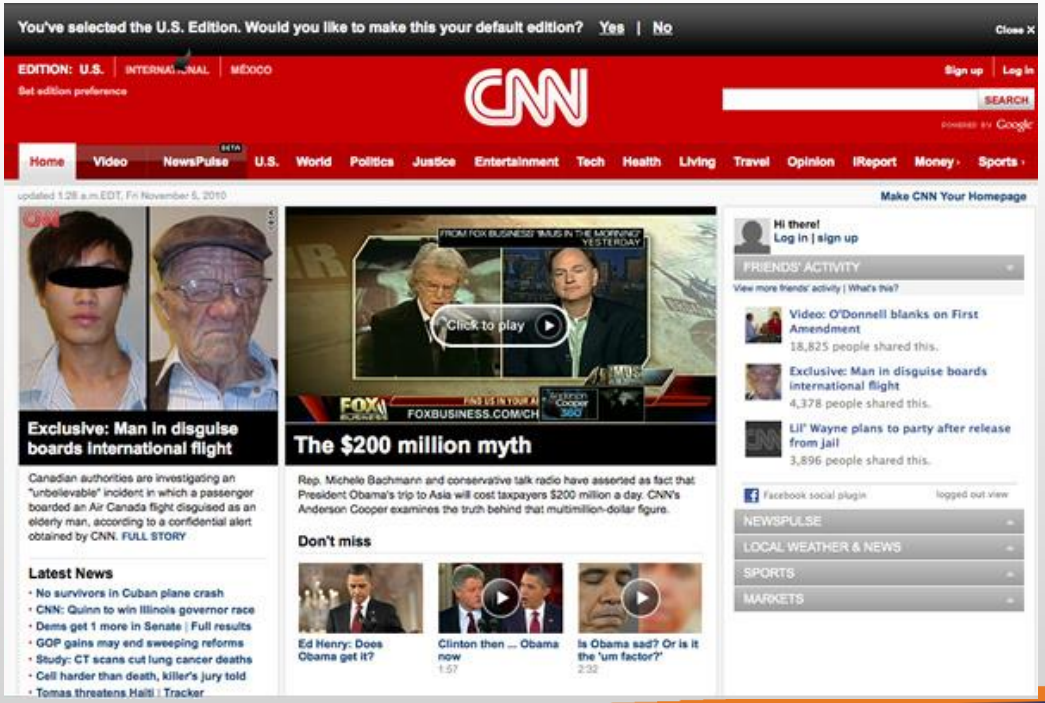
- Best practice is usually to choose a very light color for the background, and a very dark color for the text itself. This is one area where color theory is crucial to the usability of a web design; In most projects, large text areas aren't a place to try to be really creative - so keep it simple and legible.

- Along with establishing readable text, contrast can also draw the viewer's attention towards specific elements of a page. **Using a variety of contrasting colors can help focus the viewer's attention on specific page elements.**

Vibrancy

Vibrancy dictates the emotion of your design. Brighter colors lead the user to feel more energetic as a result of your design, which is particularly effective when you are trying to advertise a product or invoke an emotional response.

Darker shades relax the user, allowing their mind to focus on other things.



CNN's website features a stark red banner across the top, which leads to heightened emotions from users as they are stimulated by the vibrancy of the design (and the contrast between red, white, and black- the primary color scheme of the website).

Color Scheme

The commonly accepted color schemes:

- triadic,
- compound
- analogous

Triadic Color Scheme

Composed of 3 colors on separate ends of the color spectrum. There is a very easy way to create a Triadic color scheme:

1. Take a color wheel, and choose your base color.
2. Draw an Equilateral Triangle from this point.
3. The three points of the triangle will form your tri-color scheme.

By using an Equilateral Triangle, you can ensure the colors have equal vibrancy and compliment each other properly.

Compound Color Scheme (aka Split Complimentary)

The Compound color scheme is based on providing a range of Complementary Colors: two colors are chosen from opposite ends of the color spectrum. By doing so, the designer is allowed more freedom in their design while also benefiting from the visual appeal of complementary colors.

Analogous

Analogous color scheme is based on a careful selection of colors in the same area of the color spectrum. Usually the colors are differentiated by their vibrancy, and their contrast when compared to each other.

Two examples of an Analogous color scheme are:

1. Shades Yellow and Orange
2. A Monochromatic Selection **(Shades of a base color)**

Colour display

- Colour adds an extra dimension to an interface and can help the user understand complex information structures.
- Colour can be used to highlight exceptional events.
- Common mistakes in the use of colour in interface design include:
  - The use of colour to communicate meaning;
  - The over-use of colour in the display.

Colour Use Guidelines

- Limit the number of colours used and be conservative in their use.
- Use colour change to show a change in system status.
- Use colour coding to support the task that users are trying to perform.
- Use colour coding in a thoughtful and consistent way.
- Be careful about colour pairings.

Error Messages

- Error message design is critically important. Poor error messages can mean that a user rejects rather than accepts a system.
- Messages should be polite, concise, consistent and constructive.
- The background and experience of users should be the determining factor in message design.

System and User-Oriented Error Messages

- System-oriented message
  - Useful for technical staff
  - Detail internal states of system
  - Good for diagnostics and repair
  - Usually complete gibberish for users
- User-oriented message
  - Useful for user to fix a problem
  - Reassuring
  - Give instructions on whom to contact if appropriate

A Friendly, Helpful Message System

- If possible, show what can be done to fix problem
- Explain WHO can do WHAT to help user in trouble
- Include telephone numbers

OUT OF DISC SPACE (FSERR 46).

vs

OUT OF DISC SPACE (FSERR 46).

Check :FREE for free space

Verify :BUILD or :FILE commands for typing error;

use 32 extents if possible; check device class;

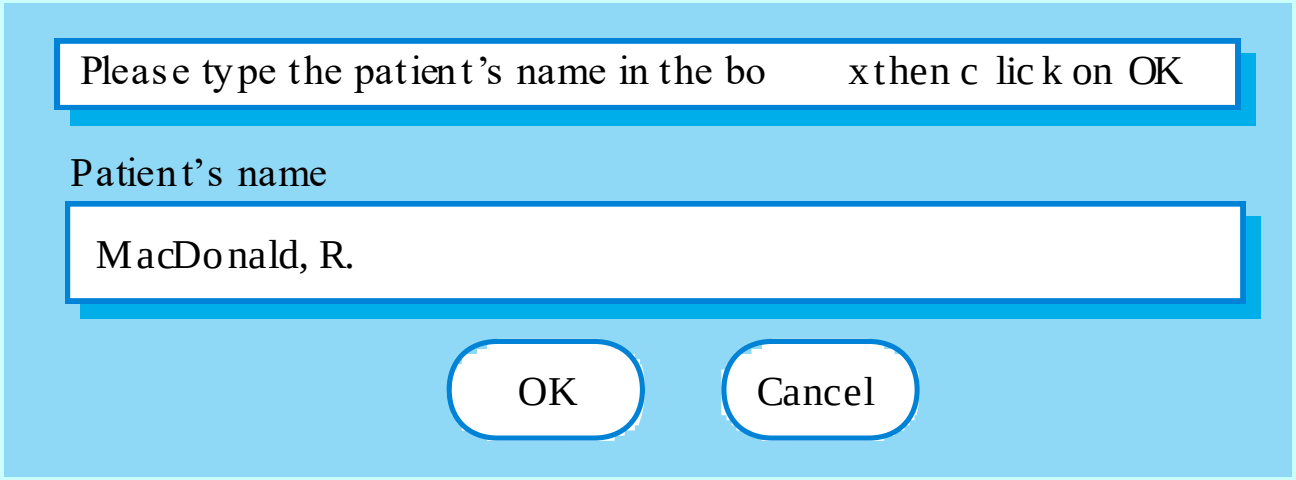
call Ramesh @ (514) 234-5678 X.216 for help

Design factors in message wording

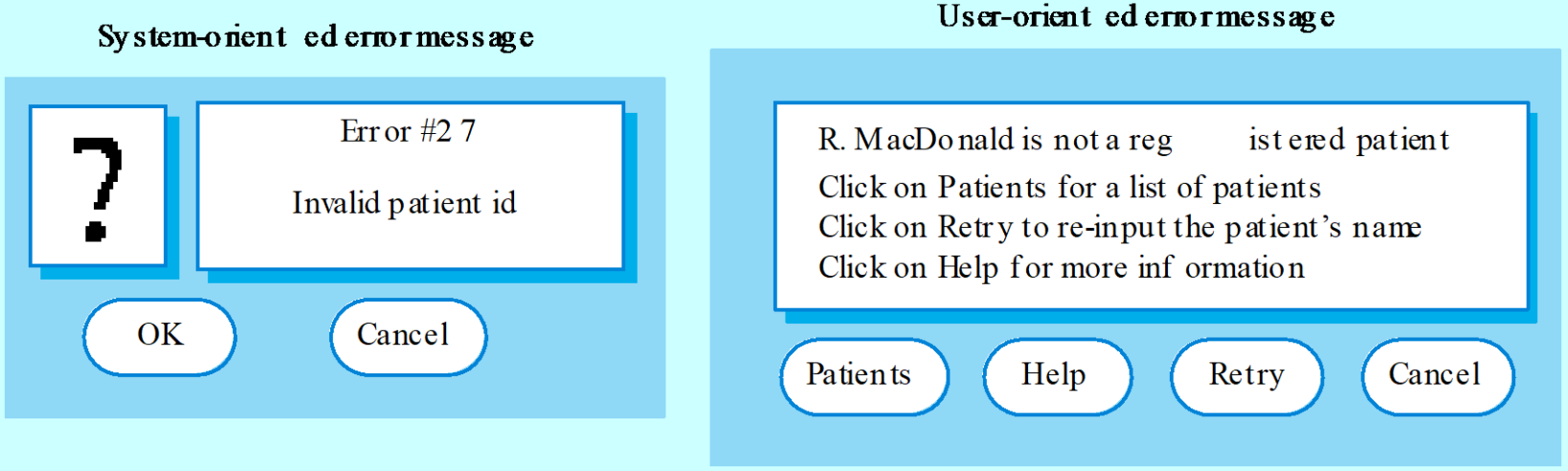
Factor	Description
Context	Wherever possible, the messages generated by the system should reflect the current user context. As far as is possible, the system should be aware of what the user is doing and should generate messages that are relevant to their current activity.
Experience	As users become familiar with a system they become irritated by long, ÕmeaningfulÕ messages. However, beginners find it difficult to understand short terse statements of a problem. You should provide both types of message and allow the user to control message conciseness.
Skill level	Messages should be tailored to the userÕsskills as well as their experience. Messages for the different classes of user may be expressed in different ways depending on the terminology that is familiar to the reader.
Style	Messages should be positive rather than negative. They should use the active rather than the passive mode of address. They should never be insulting or try to be funny.
Culture	Wherever possible, the designer of messages should be familiar with the culture of the country where the system is sold. There are distinct cultural differences between Europe, Asia and America. A suitable message for one culture might be unacceptable in another.

User error

- Assume that a nurse misspells the name of a patient whose records he is trying to retrieve.



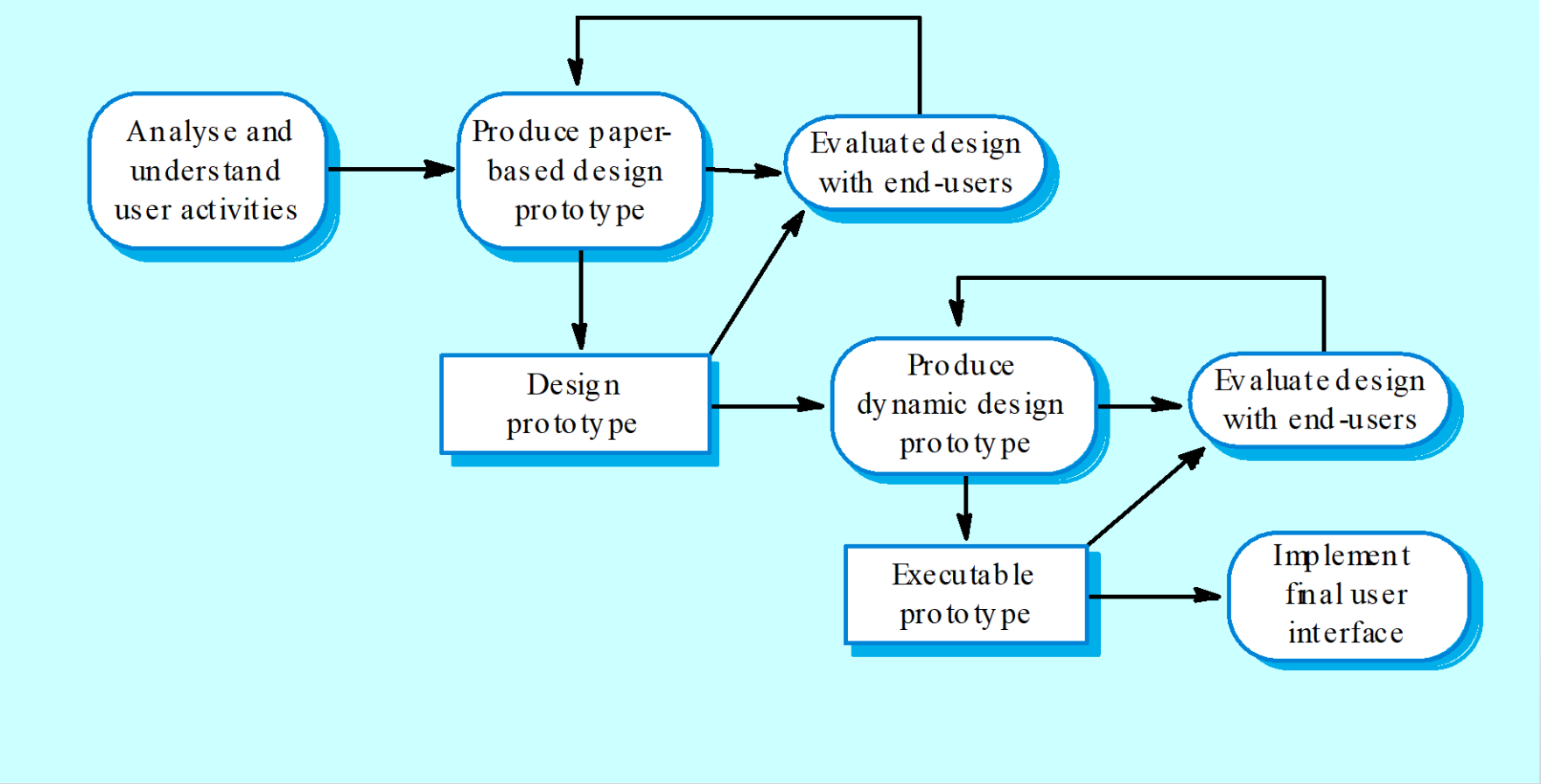
Good and bad message design



The UI design process

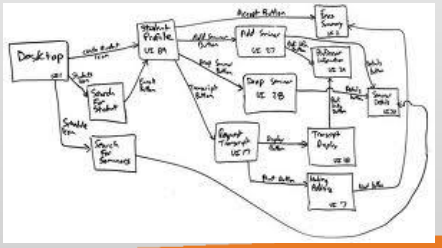
- UI design is an iterative process involving close liaisons between users and designers.
- The 3 core activities in this process are:
  - **User analysis:** Understand what the users will do with the system;
  - **System prototyping:** Develop a series of prototypes for experiment;
  - **Interface evaluation**:Experiment with these prototypes with users.

The Design Process

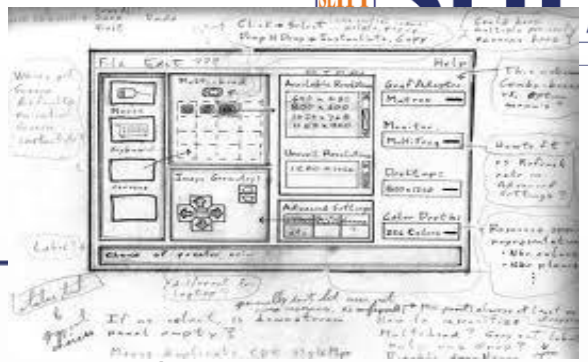


User Interface Prototyping

- The aim of prototyping is to allow users to gain direct experience with the interface.
- Without such direct experience, it is impossible to judge the usability of an interface.
- Prototyping may be a two-stage process:
  - Early in the process, paper prototypes may be used;
  - The design is then refined and increasingly sophisticated automated prototypes are then developed.



Paper Prototyping



- Work through scenarios using sketches of the interface.
- Use a storyboard to present a series of interactions with the system.
- Paper prototyping is an effective way of getting user reactions to a design proposal.

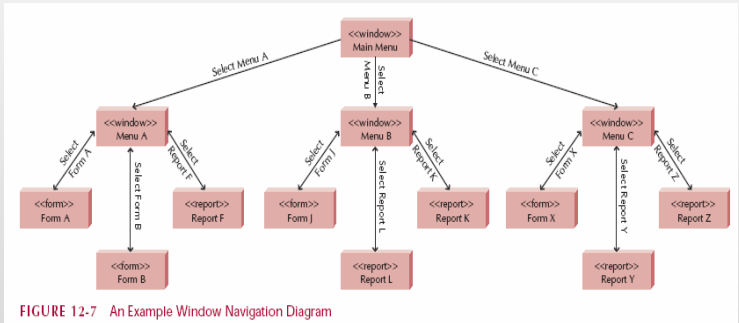
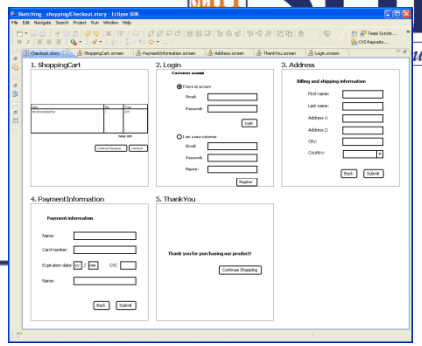


FIGURE 12-7 An Example Window Navigation Diagram

Prototyping Techniques



- Script-driven prototyping
  - Develop a set of scripts and screens using a tool such as Macromedia Director. When the user interacts with these, the screen changes to the next display.
- Visual programming
  - Use a language designed for rapid development such as Visual Basic.
- Internet-based prototyping
  - Use a web browser and associated scripts.

Visio/HTML/

User Interface Evaluation

- Some evaluation of a user interface design should be carried out to assess its suitability.
- Full scale evaluation is very expensive and impractical for most systems.
- Ideally, an interface should be evaluated against a usability specification. However, it is rare for such specifications to be produced.

Usability Attributes

Attribute	Description
Learnability	How long does it ta ke a new user to become productive with the system?
Speed of operation	How well does the system response match the userÕswork practice?
Robustness	How tolerant is the system of user error?
Recoverability	How good is the system at recovering from user errors?
Adaptability	How closely is the system tied to a single model of work?

Simple Evaluation Techniques

- Questionnaires for user feedback.
- Video recording of system use and subsequent tape evaluation.
- Tools to collect information about facility use and user errors.
- The provision of code in the software to collect on-line user feedback.

Key Points

- User interface design principles should help guide the design of user interfaces.
- Interaction styles include direct manipulation, menu systems form fill-in, command languages and natural language.
- Graphical displays should be used to present trends and approximate values. Digital displays when precision is required.
- Colour should be used sparingly and consistently.

Key Points

- The user interface design process involves user analysis, system prototyping and prototype evaluation.
- The aim of user analysis is to sensitise designers to the ways in which users actually work.
- The goals of UI evaluation are to obtain feedback on how to improve the interface design and to assess if the interface meets its usability requirements.

Any Questions ???

UI Sketching

SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY

B. SC. SPECIAL HONORS DEGREE IN INFORMATION TECHNOLOGY

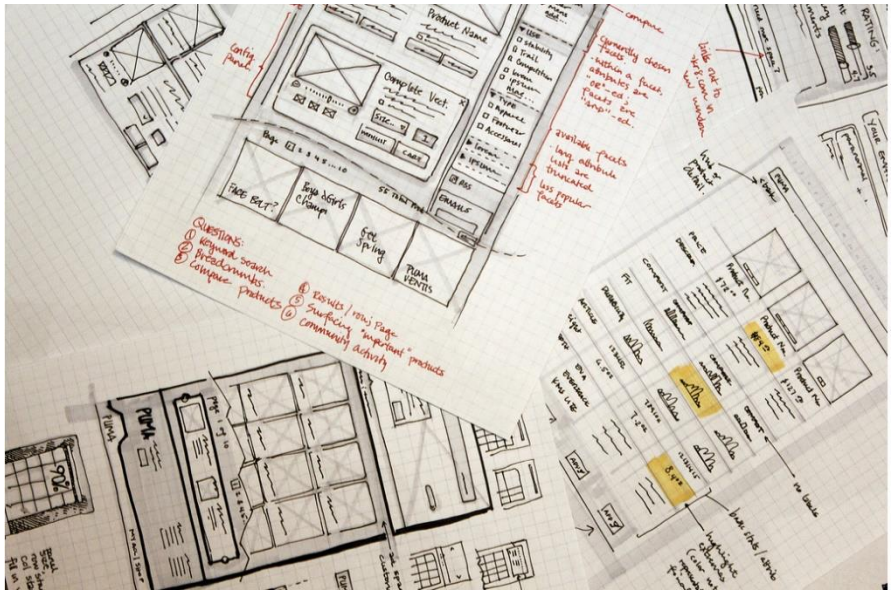
YEAR 2 – SEMESTER 2 - 2020

What is UI Sketching?

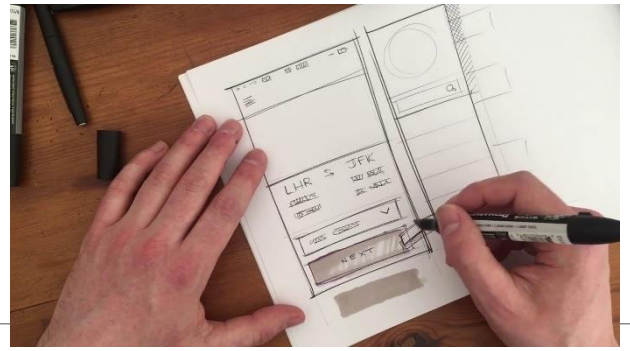
This simply refers to sketching out the basics of a user interface before getting into wire framing, prototyping and coding.

Normally this comes right after you got an idea for a new project.

This UI sketch is the first step to understand the problem properly.

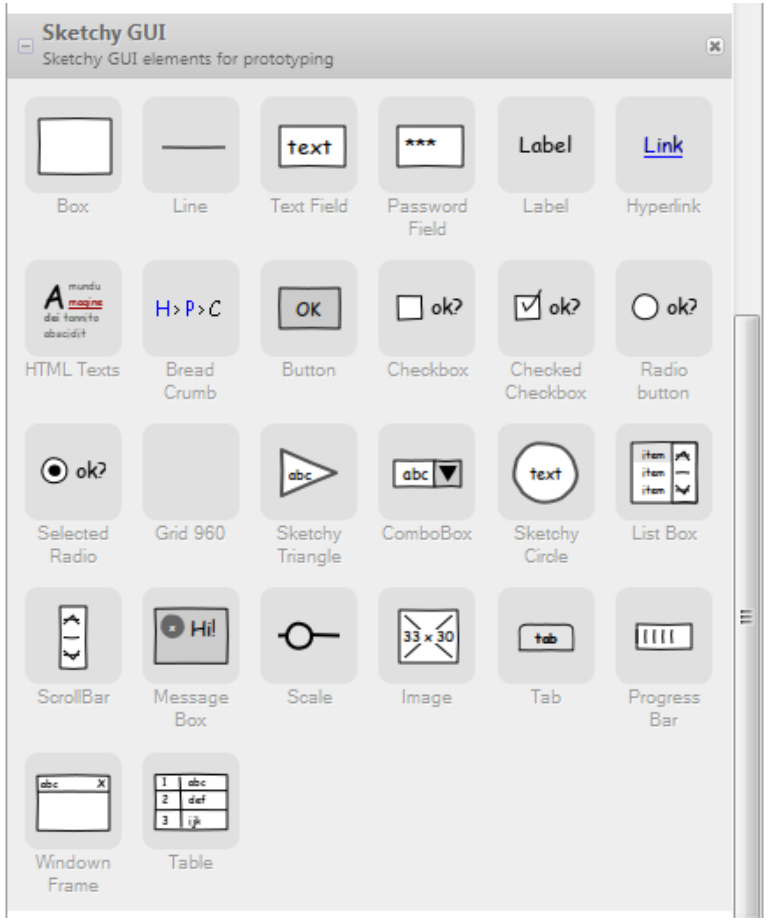


Importance of UI Sketching



- Best way to identify all the possible options and found the most effective, user-friendly way to express your interface idea.
- UI sketching is fast. There's nothing faster than drawing with a pen and paper.

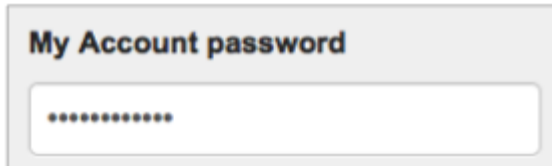
Notations for Common User Interface Elements



Input Controls

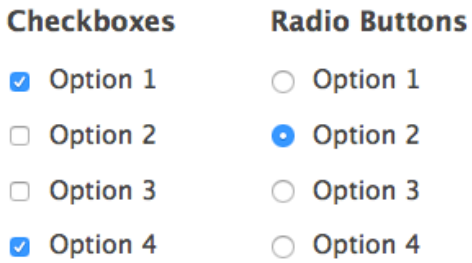
- **Text Box**

- **Password Field**



- **Checkboxes and Radio Buttons**

  - **Example:**

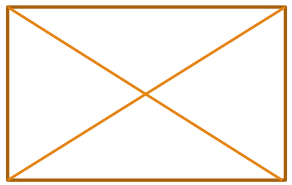


Input Controls

- Buttons

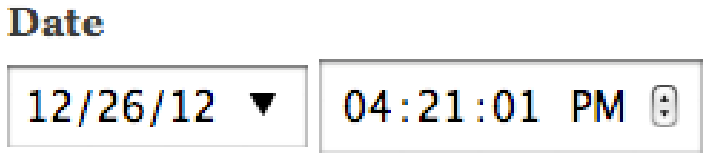
- Browse Button

- Images



- Date Picker

- Date and Time

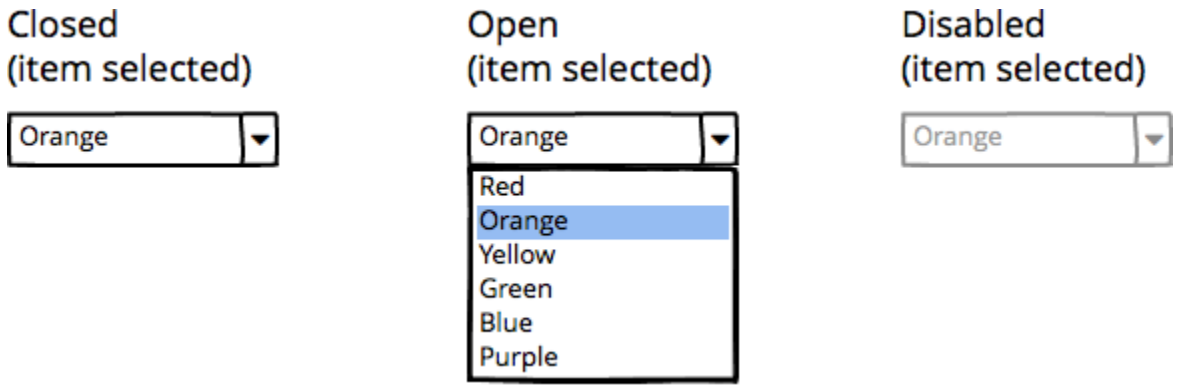


Input Controls

- **Dropdown Menu (Combo Box)**

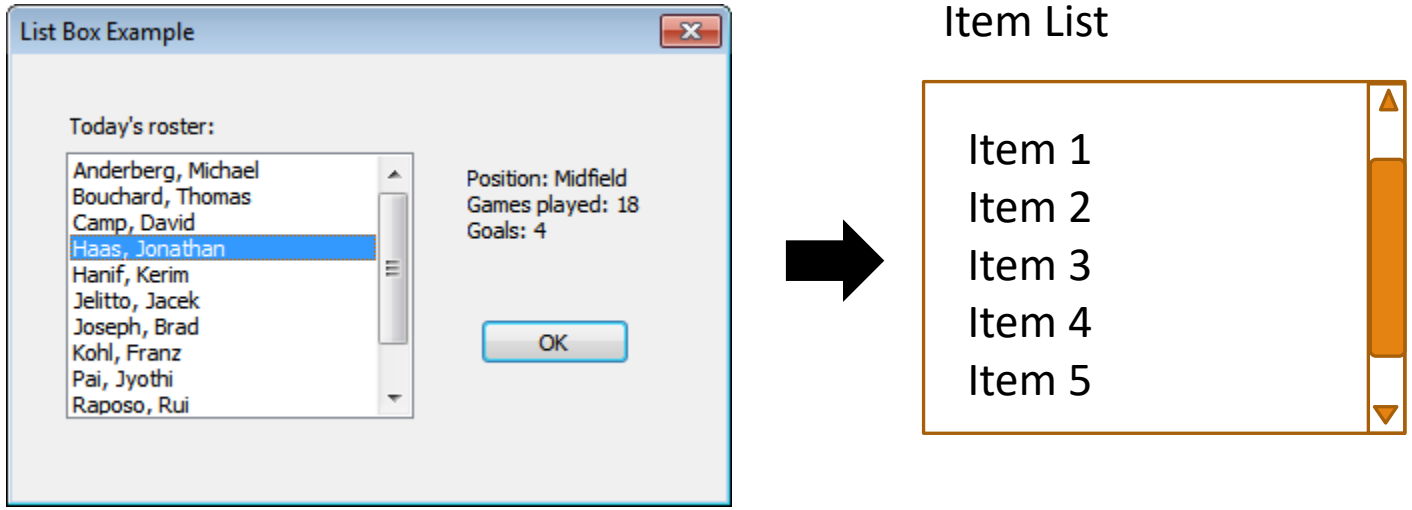
State: California

  - **Examples:**



Input Controls

List Box - Allow users to select a multiple items at a time



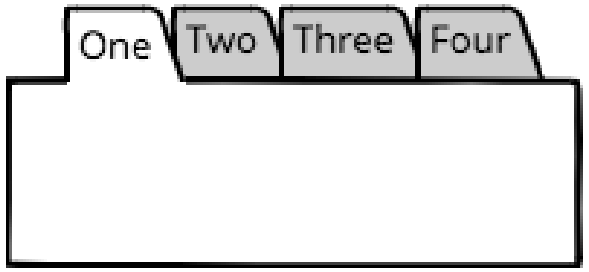
Navigation Controls

- **Links**

Default Active Visited

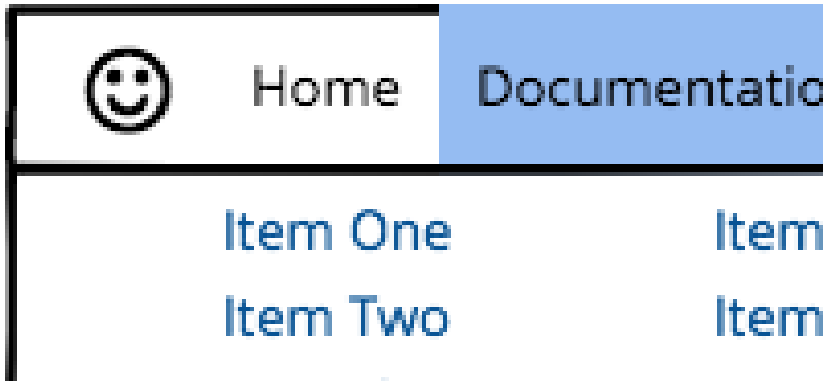
External Light on dark

- **Tabs**



Navigation Controls

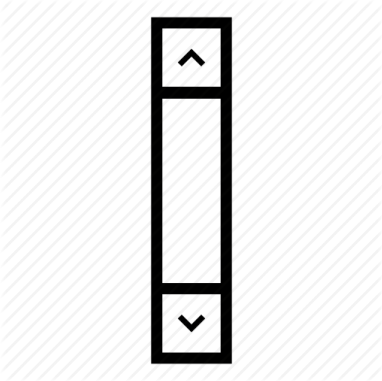
- **Menu Bars**



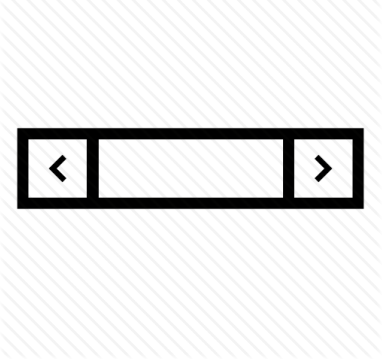
- **Breadcrumbs**

Other Controls

- **Scroll Bar**
  - **Vertical Scroll bar**



Horizontal Scroll bar



UI Sketching/Wire framing Tools

There are lot of software tools available for UI sketching. Some of them as follows;

- **Pencil Project** — A quick wireframe tool focuses on diagrams and GUI prototype (Free)
- **Mockplus** — A simple & clean tool makes you focus on the design instead of spending time on learning it. (Only Basic level is free)
- **iDoc** — A handy design tool helps create wireframes, make interactions, comment, download and handoff designs with ease. (Only Basic level is free)
- **Wireframe CC** — The minimal quick wireframe tool (Free)
- **Balsamiq Mockups** — A quick wireframe tool based on Flash (Free Trial)

References

- https://www.justinmind.com/blog/from-a-mere-wireframe-to-a-final-website-design/
- https://balsamiq.com/learn/resources/controls/

SLIIT
Discover Your Future

Software Testing

Lecture 4

Sri Lanka Institute of Information Technology
B. Sc. Special Honors Degree In Information Technology
Year 2 – Semester 2

What is testing?

- Software testing is an vital part of the software lifecycle.
- Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not.
- Software testing also identifies important defects, flaws, or errors in the application code that must be fixed

Observations about Testing

- *"... the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual results ..."*

  (ANSI/IEEE Standard 729, 1983).

- *"... the process of executing a program with the intent of finding errors..."*

  (Myers, G. J., The Art of Software Testing, Wiley, 1979).

- A software product should only be released after it has gone through a proper process of development, testing and bug fixing.
- Testing looks at areas such as performance, stability and error handling by setting up test scenarios under controlled conditions and assessing the results.
- This is why exactly any software has to be tested.
- It is important to note that software is mainly tested to see that it meets the customers' needs and that it conforms to the standards.
- It is a usual norm that software is considered of good quality if it meets the user requirements.

Who does Testing?

- Large IT companies have a team with responsibilities to evaluate the developed software in context of the given requirements.
- In most cases, the following professionals are involved in testing a system
  - ✓Software Tester
  - ✓Software Developer
  - ✓Project Lead/Manager
  - ✓ End User
- Different companies have different designations for people who test the software on the basis of their experience and knowledge such as **Software Tester, Software Quality Assurance Engineer, QA Analyst**, etc

Software testing answers questions..

- Does it really work as expected?
- Does it meet the users' requirements?
- Is it what the users expect?
- Do the users like it?
- Is it compatible with our other systems?
- How does it perform?
- How does it scale when more users are added?
- Which areas need more work?
- Is it ready for release?

What can we do with the answers to these questions?

- Save time and money by identifying defects early
- Avoid or reduce development downtime
- Provide better customer service by building a better application
- Know that we've satisfied our users' requirements
- Build a list of desired modifications and enhancements for later versions
- Identify and catalog reusable modules and components
- Identify areas where programmers and developers need training

What do we test?

- Focus on the core functionality
- the parts that are critical or popular
- Concentrate on the application's capabilities in common usage situations before going on to unlikely situations

What makes a good tester?

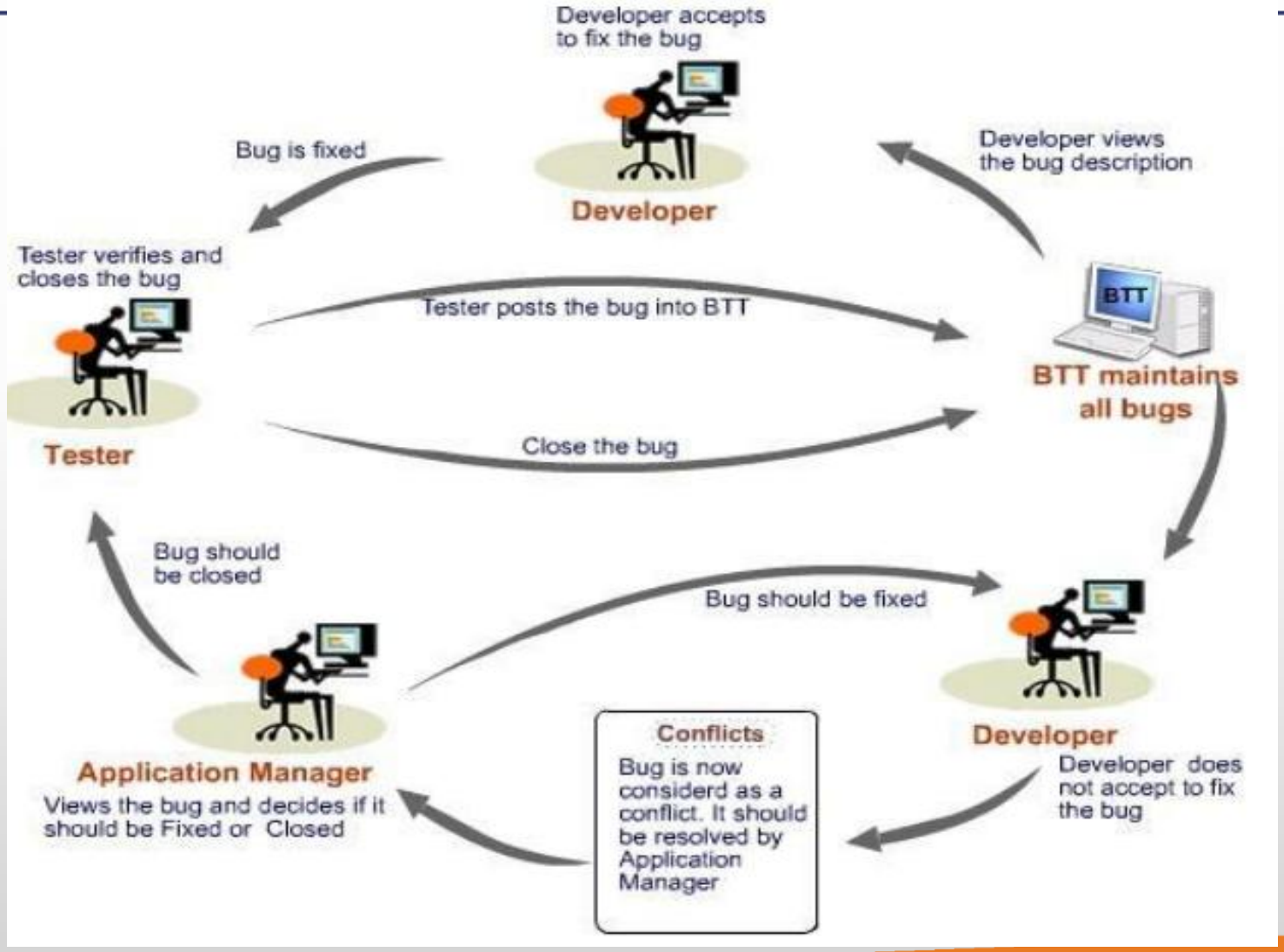
As software engineering is now being considered as a technical engineering profession, it is important that the software test engineer's posses certain traits with a relentless attitude to make them stand out. Here are a few

- ✓ **Know the technology -** Knowledge of the technology in which the application is developed is an added advantage to any tester.
- ✓ **An explorer -** bit of creativity
- ✓ **Organized-** Best testers very well realize that they too can make mistakes and don't take chances. They are very well organized and have checklists, use files, facts and figures to support their findings that can be used as an evidence and double check their findings.
- ✓ **Defects are valuable**- Good testers learn from them. Each defect is an opportunity to learn and improve. Learning from defects helps – prevention of future problems, track improvements, improve prediction and estimation

Guidelines for new testers

- **Testing can't show that bugs don't exist**.
- **It is impossible to test a program completely -** number of paths through the software is very large, and the specification is subjective to frequent changes
- **Target environment and intended end user**
- **Be the customer**
- **Build your credibility**
- **Review competitive products.**
- **Follow standards and processes -** As a tester, your need to conform to the standards and guidelines set by the organization

Bug Life Cycle and how a bug can be tracked using Bug Tracking Tools (BTT)



Testing Levels and Types

Levels of Testing

- **Unit Testing:** To verify a single program or a section of a single program
- **Integration Testing:** To verify interaction between system components.(Tests that check that modules work together in combination) Prerequisite: unit testing completed on all components
- **System Testing:** To verify and validate behaviors of the entire system.
  Have all user stories been implemented and function correctly?
- **Acceptance Tests:** Tests performed by the user to check that the delivered system meets their needs

Types of Testing

- Formal Testing
- Informal Testing
- Manual Testing
- Automated Testing
- Black box Testing
- White box Testing
- Regression Testing
- Adhoc Testing
- Load Testing
- Stress Testings
- Performance Testing
- Usability Testing
- Recovery/Error Testing
- Security Testing
- Comparison Testing
- Smoke Testing

Test Case Development

Test plan describes what to test, a test case describes how to perform a particular test.

You need to develop test cases for each test listed in the test plan.

Test case ID : Each test case should have a unique ID.	**Test designed by:** Tester's Name
Test title: Should provide a concise, revealing description of the test case, such as " Reset Pass ". The title is important because it's often the first or only thing you see when you are scanning a list of test cases. Clear titles are the key to help testers to find quickly the right test cases.	**Test designed day:** Date when test was designed
Test priority (High/Medium/Low): It is useful while executing the test.	**Test executed by:** Who executed the test(tester)
Module name:	**Test executed day:** Date when test was executed
Description: A detailed description of the test case.	
Preconditions (if there are any): Any requirement that needs to be done before execution of this test case	
Dependencies (if there are any):	

Test steps:

- Test Steps section gives the tester a numbered list of the steps to perform in the system, which makes it easier to understand the test case.
- It is recommended to have 3-8 test steps per one test case. Too many steps make it difficult for developers and testers to reproduce the steps when a bug report is filed against the test case.

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
	Inputs you are going to give (actual values)	Mention the expected result including error or message that should appear on the screen. The tester needs to know the expected result in order to assess whether the test case is successful	The output	if actual result is not the same as the expected result – fail	If there are some special conditions which is left in the above field

Test case ID :BU_001	Test designed by: Sujeepan
Test title: Test the Login Functionality in Banking	Test designed day: 03/08/2018
Test priority (High/Medium/Low): High	Test executed by: Isuru
Module name: Bank login screen	Test executed day: 08/08/2018
Description: Verify login with valid username and password	
Preconditions (if there are any): User has valid username and password	
Dependencies (if there are any):	

Test steps:

Navigate to Login Page
In the ' User Name' field, enter the username of the registered user.
In the 'Password' field, Enter the password of the registered user
Click 'Sign In' Button

Test ID	Test Inputs	Expected Output	Actual Output	Result (Pass/Fail)	Comments
BU_001	Username :suppu@gmail.com Password : 943170624V	Redirect to the Home Page	Redirect to the Home Page	Pass	
BU_001	Username : binthu@gmail.com Password :	Error message showing "Please enter the password"	Redirect to the Home Page	Fail	

Automation Testing

Automation Testing means using an automation tool to execute your test case suite

Automated software testing is important due to following reasons:

- Manual Testing of all workflows, time and money consuming
- It is difficult to test for multilingual sites manually
- Automation does not require Human intervention. You can run automated test unattended (overnight)
- Automation increases the speed of test execution
- Automation helps increase Test Coverage
- Manual Testing can become boring and hence error-prone.

Automation Testing Tools

Product	Selenium	Katalon Studio	Unified Functional Testing	TestComplete	watir
Available since	2004	2015	1998	1999	2008
Application Under Test	Web apps	Web (UI & API), Mobile apps	Web (UI &API), Mobile, Desktop, Packaged apps	Web (UI &API), Mobile, Desktop apps	Web apps
Pricing	Free	Free	$$$$	$$	Free
Supported Platforms	Windows Linux OS X	Windows Linux OS X	Windows	Windows	Windows Linux OS X
Scripting languages	Java, C#, Perl, Python, JavaScript, Ruby, PHP	Java/Groovy	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++ and C#	Ruby
Programming skills	Advanced skills needed to integrate various tools	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts	Advanced skills needed to integrate various tools
Ease of Installation and Use	Require advanced skills to install and use	Easy to setup and use	Complex in installation. Need training to properly use the tool	Easy to setup. Need training to properly use the tool	Advanced skills needed to integrate various tools

Few online courses

- https://www.coursera.org/learn/uva-darden-agile-testing
- https://www.edx.org/course/software-testing-management-usmx-umuc-stv1-2x-1
- https://www.edx.org/course/formal-software-verification

Following are sample required

Required Skills & Technical Experience

- A University degree in IT, computer science, computer engineering or an equivalent qualification.
- 1+ years of experience working with IT projects involving full test development life cycle.
- Extensive experience in functional & non-functional testing types.
- Hands on experience in Agile project environments and Agile processes.
- Exposure in designing automation test frame works and writing automation test scripts for web applications and mobile applications.
- Knowledge about RDBMS, having experience writing SQL scripts and manipulating data.
- Good problem solving and analytical skills.
- Excellent communication skills.

Internship

We're looking for a Quality Assurance Intern to join our team. You will be working closely with the Product Manager and Senior Development team and test various scenarios both through use cases, third party user testing, and in-house user testing.
Responsibilities

- Define test plans and test specifications for functional, integration and regression testing, hands-on execution of test cases, and reporting of software failures.
- Assisting in the design and evolution of the company's QA process.
- Planning, creating, executing, and documenting automated test scripts for projects.
- Identifying and communicating risk and risk mitigation strategies.
- Setting and meeting schedule estimates, timelines, and milestones.
- Continuous improvement over test coverage - review bugs reported by clients or client services, and incorporate learned lessons into test plans.

To qualify for this position, the following may be required:

- Strong understanding of HTML, CSS, Javascript, Java, and PHP.
- Documentation skills to accurately represent a complex knowledge set

Completion a Bachelors of Science degree in Computer Science or Computer Engineering from an accredited four-year university is a plus.