

# Writing in the Field of Computer Science

Dr. Wedad Hussein

Information Systems Department

wedad.hussein@fcis.asu.edu.eg



# Writing Project Proposals

# Why it's Needed

- To get approval.
- To get sponsoring.
- To get funding.

You need to convince someone.

# General Points to Cover

- Introduce yourself and your project.
- Describe the need and how the project will meet that need.
- Provide the details of what you propose to do and explain the costs.
- Persuade your readers that you are the perfect choice to successfully complete the project.

# Know Your Audience

- Any proposal reader will want to know why you are proposing the project to them.
- You need to convince the readers that it's in their best interest to support your project.
- You need to write a customized proposal.

# Cover Letter

- You may need to start your proposal with a Cover Letter:
  - ◆ A brief personal introduction of yourself and your project.
  - ◆ The action you want them to take after reading your proposal.
  - ◆ Contact information.

# Suggested Content

## 1. Introduction:

- ◆ What is the history of the problem?
- ◆ Why is this problem interesting?
- ◆ Is the problem already solved? What is done now?
- ◆ Are there any similar systems or solutions to the one you propose?
- ◆ Are there are possible improvements to current solutions?

# Suggested Content (cont.)

## 2. Project Summary:

- ◆ What in general will this project achieve?

## 3. Project Details

- ◆ *Architecture and Environment:* Describe the project environment (software, hardware, languages, organizations, etc.)
- ◆ *Implementation Issues and Challenges:*
  - What will be the most difficult issues and challenges in the implementation?
  - How are you using or extending current tools/systems for your problem?
  - What makes your project unique?



# Suggested Content (cont.)

## 3. Project Details (cont.)

### ◆ *Deliverables:*

- What will the project produce? (program, report, etc.)
- Describe in relative detail the features of each of the project's products.
- Emphasize what your project contributes or achieves.

### ◆ *Time Plan:*

- Provide an estimated timeline of project deliverables and important dates.

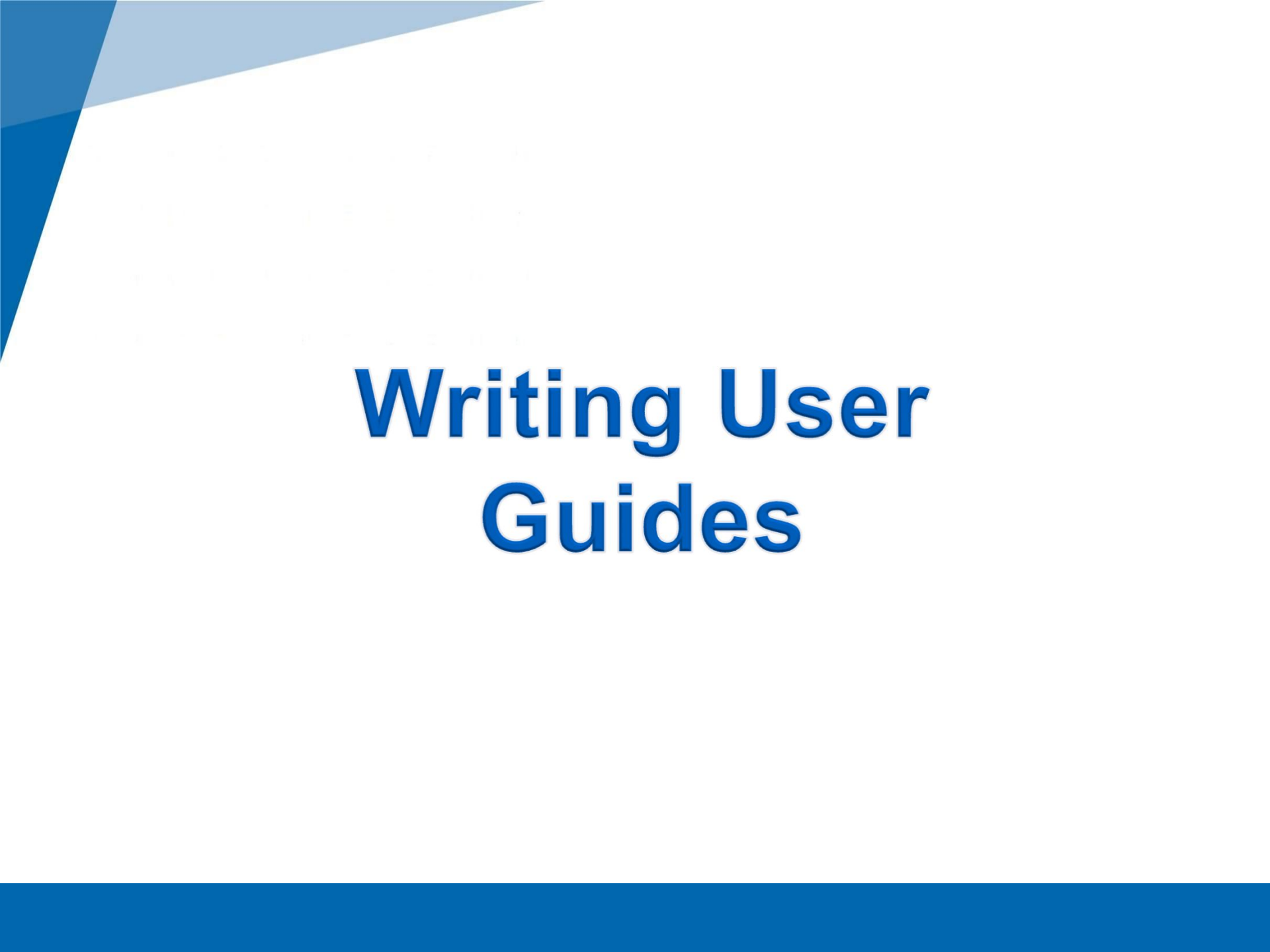
# Suggested Content (cont.)

## 4. Conclusion:

- ◆ Summarize the project including the problem, motivation, and proposed solution, and re-state important (planned) contributions.

## 5. References

- ◆ List references used to compile proposal and references that will be used for project (if already known).



# Writing User Guides

# User Guide

A User Guide explains how to use a software application in language that a non-technical person can understand.

# Identify the Audience

- Who are your users?
- Are there different groups of users?
- Will different groups of users perform different tasks?
- What level of technical expertise do users have?
- How much time will they invest reading the user guide?
- What tasks are users typically going to perform with the software?

# Content

## 1. Preface:

- ◆ Use this section to reference other documents related to the software.
- ◆ If needed include a “How to use this guide” section

## 2. Table of contents.

## 3. Body

## Content (cont.)

### 4. Reference Materials:

- ◆ Error messages that may arise when you use the application.
- ◆ Troubleshooting tips to resolve these issues.
- ◆ Frequently asked questions.

### 5. Glossary: covers all acronyms and terms used in the document.

### 6. Index: helps users locate specific items very fast without having to search through the entire document manually.

# Procedures

- Procedures help the user perform specific tasks.
- Examples:
  - ◆ When, why, and how you can perform a task.
  - ◆ What the screen will show after you perform a task.
  - ◆ Examples of tasks and program operation.



# Writing Procedures

- Tasks include:
  - ◆ Identifying the major tasks.
  - ◆ Separating each major task into subtasks.
  - ◆ Writing a series of steps that walk the user through each subtask.
  - ◆ Using an "if-then" approach when explaining decisions that users can make.

# Guidelines

- Make sure the instructions actually map on to the product in all respects.
- Include a one-page quick start guide.
- Present instructions as step-by-step procedures.
- Tell the user what functions there are, and what they are for — not just how to use them.
- Avoid marketing.

## Guidelines (cont.)

- Avoid lengthy paragraphs.
- Use everyday words and terms.
- Explain symbols, icons and codes early.
- Avoid creating dead-ends.
- Do not assume the user has prior experience or product knowledge.
- Write in the present tense and the active voice.



# Writing Code Comments

# Why is it Important

- The lifetime of a software is 10% development and 90% maintenance.
- Maintenance is where comments can be useful.
- Developers don't stay for the whole life cycle.
- Lack of comments will eventually lead to lost productivity due to time spent tracing and re-learning.

# Guidelines

- Don't rely on comments, write readable code.
- Don't write what the code is doing.
- Always write why you are writing this piece of code.
- One liner comment is best.

## Tips

- Don't comment the obvious.



```
//The width of the line is 2  
lineWidth = 2;
```

- Write in English.
- Both writing no comment and writing too much comment is bad.

