ESLSCA University

This is my title

by

My Name
My Teammate

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Computing & Digital Technology (Media Informatics and Mixed Reality Program)

in the

School Of Computing & Digital Tech
of the
ESLSCA University, EGYPT

Graduation Projects advisor:
Dr. ABC
Dr. XYZ

(July 2024)

Abstract

Gestures have a lot of potentials as a natural interaction method that can enrich the interaction between humans and ubiquitous environments. The lack of ordinary devices in ubiquitous environments like the keyboard and mouse make researchers work on utilizing hand gestures for interaction. However, hand gestures could change according to the situation it is performed.

Acknowledgments

I am heartily thankful to Professor

Contents

List of Tables

List of Figures

Chapter 1

Introduction

You will find many topics here to put from the Proposal

1.1 Introduction guide lines

A good introduction should tell the reader what the project is about without assuming special knowledge and without introducing any specific material that might obscure the overview. It should anticipate and combine main points described in more detail in the rest of the project report. Also, importantly, it should enthuse the reader about the project, to encourage them to read the whole report. Normally it should include such things as:

- the aim(s) or goal(s) of the project
- the intended audience or "beneficiaries" of the work done and the scope of the project"
- the approach used in carrying out the project
- assumptions on which the work is based; and
- a broad summary of important outcomes.

1.2 Problem Definition

Please put a focus on 1 2 challenges that this project aims to solve and state them very clear as your formal problem statement.

1.3 Motivation

Why Did you choose this idea ? Discus as much as market needs and academic needs for your project. It is expected that in this section you will get about 10 15 general technical CS challenges. Use surveys for showing the market need of your project. Why this system is to be developed. Your goals behind working and achieving your project

1.4 Project Description

Show down with a figure the proposed system.

1.5 Scope

The scope of your project.

Chapter 2

Background

2.1 General Guidelines

The purpose of the Background section is to provide the typical reader with information that they cannot be expected to know, but which they will need to know in order to fully understand and appreciate the rest of the report It should explain why the project is addressing the problem described in the report, indicate an awareness of other work relevant to this problem and show clearly that the problem has not been solved by anyone else. This section may describe such things as:

- The wider context of the project;
- The problem that has been identified
- Likely stakeholders within the problem area
- Any theory associated with the problem area
- Any constraints on the approach to be adopted
- Existing solutions relevant to the problem area, and why these are unsuitable or Insufficient in this particular case Methods and tools that your solution may be based on or use to solve the problem and so on.

The wider context of the project includes such things as its non-computing aspects. So, for example, if you are producing software or any other products, including business recommendations, for a specific organisation then you should describe aspects of that organisations business that are relevant to the project.

2.2 Literature Review

List down at least 10 papers from ACM and IEEE for similar work experience in the domain of your problem. You can add 5 papers or more from other sources (Springer, Elsevier, Website ..etc)

Be sure that each paper you list include the following points

1. Motivation of this work (Why the researchers do it)
2. The main problem statement of the work.
3. How the researchers contributed to solve the problem
4. What main results the researchers reach.
5. How do you think this paper you read is important for you.

2.3 Example

This is Example for writing some related work you can extract them from your Proposal and SRS

The study of gesture recognition with a presentation viewer application was shown in [1]. They show an active region for starting and ending gesture interaction. Also, they point out that gestures can be useful in crowded or noisy situations, such as in a stock exchange or manufacturing environment. Head and hand gestures have been used for limited interactions as demonstrated in by Keates et al. [1]. They discussed the problem of learning gestures and showed the importance of customization. Kurze et al. [2] presented penalization of multi-modal applications as a design approach. They focus on implicit and explicit customization of systems according to a user's preferences. Kawsar et al. [2] presented customizing the proactive applications preferences in a ubiquitous environment. They present customization in many levels of artifact, action, interaction, and timing preferences.

2.4 Comparison with Proposed Project

2.5 Screen Shots from previous systems (if needed)

2.6 Project Management and Deliverable

2.6.1 Tasks and Time Plan

Use some software for primitive plan of your project.

2.6.2 Budget and Resource Costs

Chapter 3

Specification - (SRS)

3.1 General Guide Lines

If you are SE student put your SRS Here

Some students can choose between those methods based on project type

1. (Application oriented) Selection of Approach
2. (Comparing Algorithm) Description of Algorithms
3. (Novel Algorithm) Problem statement

A specification should tell the reader what the software system is required to do.

Describing what a software system does (specification) and how it does so (design) effectively usually means describing it from more than one viewpoint. Each viewpoint will convey some information about the system that other viewpoints omit.

Possible viewpoints might be:

- The business model the software supports;
- The user interface;
- The dynamic behaviour of the system;
- How data flows through the system;
- What data types are implemented in the system;
- What algorithms are implemented in the system;
- The static architecture of the system, i.e. how the code is partitioned into modules, etc.

A common approach is to first define the user or business requirements, then describe the static architecture, identify modules and groups of closely connected modules, and then to apply other views to each of these groups. Fine details, specifically details of code, should be left out. We strongly recommend that you make extensive use of diagrams, such as entity-relationship diagrams, UML diagrams, state charts, or other pictorial techniques

3.2 Introduction

3.2.1 Purpose of this document

Describes the purpose of the document, and the intended audience.

3.2.2 Scope of this document

Describes the scope of this requirements definition effort. Introduces the requirements elicitation team, including users, customers, system engineers, and developers. This section also details any constraints that were placed upon the requirements elicitation process, such as schedules, costs, or the software engineering environment used to develop requirements.

3.2.3 Overview

Provides a brief overview of the product defined as a result of the requirements elicitation process.

3.2.4 Business Context

Provides an overview of the business organization sponsoring the development of this product. This overview should include the business's mission statement and its organizational objectives or goals.

3.3 General Description

3.3.1 Product Functions

Describes the general functionality of the product, which will be discussed in more detail below.

3.3.2 Similar System Information

Describes the relationship of this product with any other products. Specifies if this product is intended to be stand-alone, or else used as a component of a larger product. If the latter, this section discusses the relationship of this product to the larger product. This is how you can [1] a document.

Table 3.1: Functional Requirement XYZ

Function Name	Short, imperative sentence stating highest ranked functional requirement.
Description	A full description of the requirement.
Critically	Describes how essential this requirement is to the overall system.
Technical issues	Describes any design or implementation issues involved in satisfying this requirement.
Cost and schedule	Describes the relative or absolute costs associated with this issue.
Risks	Describes the circumstances under which this requirement might not able to be satisfied, and what actions can be taken to reduce the probability of this occurrence.
Dependencies with other requirements	Describes interactions with other requirements.
Pre-Condition	The state of the system before the running of the function
Post-Condition	The state of the system after the run of the function

3.3.3 User Characteristics

Describes the features of the user community, including their expected expertise with software systems and the application domain.

3.3.4 User Problem Statement

This section describes the essential problem(s) currently confronted by the user community.

3.3.5 User Objectives

This section describes the set of objectives and requirements for the system from the user's perspective. It may include a "wish list" of desirable characteristics, along with more feasible solutions that are in line with the business objectives.

3.3.6 General Constraints

Lists general constraints placed upon the design team, including speed requirements, industry protocols, hardware platforms, and so forth.

3.4 Functional Requirements

This section lists the functional requirements in ranked order. Functional requirements describes the possible effects of a software system, in other words, what the system must accomplish. Other kinds of requirements (such as interface requirements, performance requirements, or reliability requirements) describe how the system accomplishes its functional requirements see table 3.1. Each functional requirement should be specified in a format similar to the following:

3.5 Interface Requirements

This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams, and so forth.

3.5.1 User Interfaces

Use some software for primitive plan of your project. Describes how this product interfaces with the user.

3.5.1.1 GUI

Describes the graphical user interface if present. This section should include a set of screen dumps or mock-ups to illustrate user interface features. If the system is menu-driven, a description of all menus and their components should be provided.

3.5.1.2 CLI

Describes the command-line interface if present. For each command, a description of all arguments and example values and invocations should be provided.

3.5.1.3 API

Describes the application programming interface, if present. For each public interface function, the name, arguments, return values, examples of invocation, and interactions with other functions should be provided.

3.5.1.4 Diagnostics or ROM

Describes how to obtain debugging information or other diagnostic data.

3.5.2 Hardware Interfaces

Describes interfaces to hardware devices.

3.5.3 Communications Interfaces

Describes network interfaces.

3.5.4 Software Interfaces

Describes any remaining software interfaces not included above.

3.6 Performance Requirements

Specifies speed and memory requirements.

3.7 Design Constraints

Specifies any constraints for the design team using this document.

3.7.1 Standards Compliance

3.7.2 Hardware Limitations

3.7.3 others as appropriate

3.8 Other non-functional attributes

Specifies any other particular non-functional attributes required by the system. Examples are provided below.

3.8.1 Security

3.8.2 Binary Compatibility

3.8.3 Reliability

3.8.4 Maintainability

3.8.5 Portability

3.8.6 Extensibility

3.8.7 Re-usability

3.8.8 Application Affinity/Compatibility

3.8.9 Resource Utilization

3.8.10 Serviceability

3.8.11 others as appropriate

3.9 Preliminary Object-Oriented Domain Analysis

This section presents a list of the fundamental objects that must be modeled within the system to satisfy its requirements. The purpose is to provide an alternative, "structural" view on the requirements stated above and how they might be satisfied in the system. A primitive class diagram to be delivered.

3.9.1 Inheritance Relationships

This section should contain a set of graphs that illustrate the primary inheritance hierarchy (is-kind-of) for the system. For example:

3.9.2 Class descriptions

This section presents a more detailed description of each class identified during the OO Domain Analysis. For more details on the process giving rise to these descriptions, see Lecture 5.3: OO Domain Analysis and/or texts on object-oriented software development. Each class description should conform to the following structure:

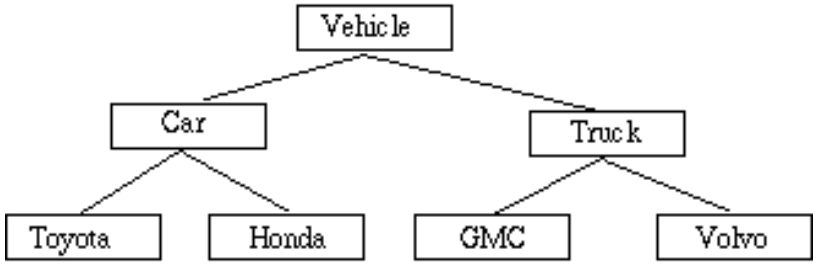

Figure 3.1: Inheritance Relations

3.9.2.1 Class name

Abstract or Concrete: Indicates whether this class is abstract or concrete.

3.9.2.2 List of Superclasses:

Names all immediate superclasses.

3.9.2.3 List of Subclasses:

Names all immediate subclasses.

3.9.2.4 Purpose:

States the basic purpose of the class.

3.9.2.5 Collaborations:

Names each class with which this class must interact in order to accomplish its purpose, and how.

3.9.2.6 Attributes:

Lists each attribute (state variable) associated with each instance of this class, and indicates examples of possible values (or a range).

3.9.2.7 Operations

: Lists each operation that can be invoked upon instances of this class. For each operation, the arguments (and their type), the return value (and its type), and any side effects of the operation should be specified.

3.9.2.8 Constraints:

Lists any restrictions upon the general state or behavior of instances of this class.

3.10 Operational Scenarios

This section should describe a set of scenarios that illustrate, from the user's perspective, what will be experienced when utilizing the system under various situations. In the article Inquiry-Based Requirements Analysis (IEEE Software, March 1994), scenarios are defined as follows: In the broad sense, a scenario is simply a proposed specific use of the system. More specifically, a scenario is a description of one or more end-to-end transactions involving the required system and its environment. Scenarios can be documented in different ways, depending up on the level of detail needed. The simplest form is a use case, which consists merely of a short description with a number attached. More detailed forms are called scripts. These are usually represented as tables or diagrams and involved identifying an action and the agent (doer) of the action. For this reason, a script can also be called an action table. Although scenarios are useful in acquiring and validating requirements, they are not themselves requirements, because the describe the system's behavior only in specific situations; a specification, on the other hand, describes what the system should do in general.

3.11 Preliminary Schedule Adjusted

This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependence's, and their tentative start/stop dates. The plan also includes information on hardware, software, and resource requirements. The project plan should be accompanied by one or more PERT or GANTT charts.

3.12 Preliminary Budget Adjusted

This section provides an initial budget for the project, itemized by cost factor.

3.13 Appendices

Specifies other useful information for understanding the requirements. All SRS documents should include at least the following two appendices:

3.13.1 Definitions, Acronyms, Abbreviations

Provides definitions of unfamiliar definitions, terms, and acronyms.

Chapter 4

Design

4.1 Design Guidelines

If your are SE Student just put your SDD

Some Students can choose between

1. (Application oriented) Application of Selected Approach
2. (Comparing Algorithm) Implementation
3. (Novel Algorithm) Alternative Designs and Final Algorithm

The design then gives the top-level details of how the software system meets the requirement. It will also identify constraints on the software solution, that are important in guiding decision making throughout the development process.

As well as describing the system, it is important that you justify its design, for example, by discussing the implications of constraints on your solution and different design choices, and then giving reasons for making the choices you did. Typically these implications will relate to the aims of the project and to aspects of it discussed in the Background section. The design of the system will almost certainly have evolved while you were developing it. Obviously you should describe its final state but often there are good reasons for describing intermediate states, too; for example, if you want to discuss the details of the design method used or to highlight learning that you later refer to in the Reflection section. If you do this, take special care to make sure the reader does not get confused between different stages of the design.

If you are not designing a system, but testing a hypothesis for a more scientifically oriented project , specification and design sections may not be required in quite the

The specification instead becomes a description of the problem and what is required of a solution. The design becomes a description of your approach to solving the problem and your suggested solutions.

For instance, if you are designing an algorithm to solve a particular problem you would have a problem statement section and then a section describing one or more suggested algorithms to solve the problem.

4.2 Introduction

4.2.1 Purpose

Identify the purpose of this SDD and its intended audience. (e.g. This software design document describes the architecture and system design of XX..).

4.2.2 Scope

Provide a description and scope of the software and explain the goals, objectives and benefits of your project. This will provide the basis for the brief description of your product.

4.2.3 Overview

Provide an overview of this document and its organization.

4.2.4 Reference Material

This section is optional. List any documents, if any, which were used as sources of information for the test plan.

4.2.5 Definitions and Acronyms

This section is optional. Provide definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD. These definitions should be items used in the SDD that are most likely not known to the audience.

Term	Definition
Software Design Document (SDD)	Used as the primary medium for communicating software design information.
Design Entity	An element of a design that is structurally and functionally distinct from other elements.

4.3 System Overview

Give a general description of the functionality, context and design of your project. Provide any background information if necessary.

4.4 System Architecture

4.4.1 Architectural Design

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems. Identify each high level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. Don t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together. Provide a diagram showing the major subsystems and data repositories and their interconnections. Describe the diagram if required.

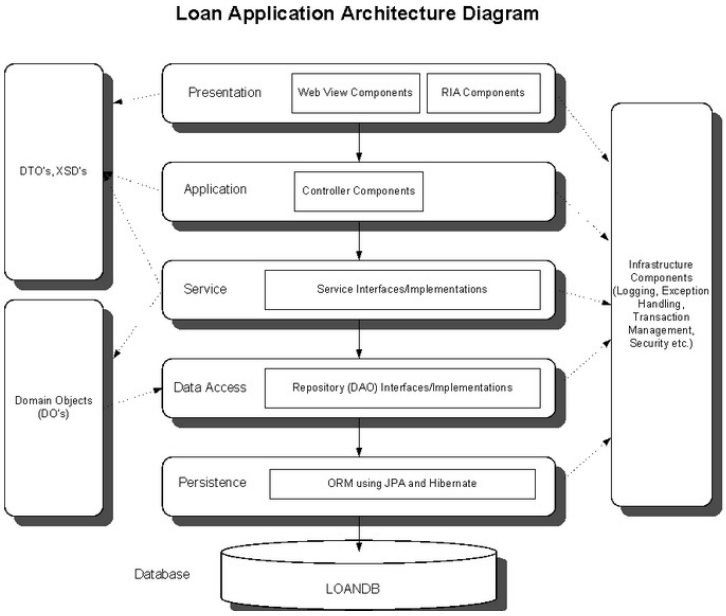

Figure 4.1: Architectural Design

4.4.2 Decomposition Description

Provide a decomposition of the subsystems in the architectural design. Supplement with text as needed. You may choose to give a functional description or an object oriented description. For a functional description, put top level data flow diagram (DFD) and

structural decomposition diagrams. For an OO description, put subsystem model, object diagrams, generalization hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), interface specifications, and sequence diagrams here.

4.4.3 Design Rationale

Discuss the rationale for selecting the architecture described in 3.1 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn't choose them.

4.5 Data Design

4.5.1 Data Description

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

4.5.2 Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

4.6 Component Design

In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudo-code. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudo code. Describe any local data when necessary.

4.7 Humnan Interface Design

4.7.1 Overview of User Interface

Describe the functionality of the system from the user s perspective. Explain how the user will be able to use your system to complete al the expected features and the feedback information that will be displayed for the user.

4.7.2 Screen Images

Display screenshots showing the interface from the users perspective. These can be hand drawn or you can use an automated drawing tool. Just make them as accurate as possible.

4.7.3 Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

4.8 Requirements Matrix

Provide a cross reference that traces components and data structures to the requirements in your SRS document. Use a tabular format to show which system components satisfy each of the functional requirements from the SRS. Refer to the functional requirements by the numbers/codes that you gave them in the SRS.

Chapter 5

Implementation

5.1 General Guidelines

Students can choose between

1. Illustrate your implementation parts
2. (Application oriented) "Deliverables" from Selected Approach
3. (Comparing Algorithm) Experiment Design

The Implementation section is similar to the Specification and Design section in that it describes the system, but it does so at a finer level of detail, down to the code level. This section is about the realisation of the concepts and ideas developed earlier. It can also describe any problems that may have arisen during implementation and how you dealt with them. Do not attempt to describe all the code in the system, and do not include large pieces of code in this section. Complete source code should be provided separately. Instead pick out and describe just the pieces of code which, for example:

- Are especially critical to the operation of the system;
- You feel might be of particular interest to the reader for some reason;
- Illustrate a non-standard or innovative way of implementing an algorithm, data structure, etc..
- You should also mention any unforeseen problems you encountered when implementing the system and how and to what extent you overcame them. Common problems are:

- Difficulties involving existing software, because of, e.g., its complexity, lack of documentation; lack of suitable supporting software;

A seemingly disproportionate amount of project time can be taken up in dealing with such problems. The Implementation section gives you the opportunity to show where that time has gone

Chapter 6

Results and Evaluation

6.1 General Rules

In this section you should describe to what extent you achieved your goals. You should describe how you demonstrated that the system works as intended (or not, as the case may be). Include comprehensible summaries of the results of all critical tests that were carried out. You might not have had the time to carry out any full rigorous tests – you may not even got as far as producing a testable system. However, you should try to indicate how confident you are about whatever you have produced, and also suggest what tests would be required to gain further confidence. This is also the place to describe the reasoning behind the tests to evaluate your results, what tests to execute, what the results show and why to execute these tests. It may also contain a discussion of how you are designing your experiments to verify the hypothesis of a more scientifically oriented project. E.g., describe how you compare the performance of your algorithm to other algorithms to indicate better performance and why this is a sound approach. Then summarise the results of the tests or experiments.

You must also critically evaluate your results in the light of these tests, describing its strengths and weaknesses. Ideas for improving it can be carried over into the Future Work section. Remember: no project is perfect, and even a project that has failed to deliver what was intended can achieve a good pass mark, if it is clear that you have learned from the mistakes and difficulties. This section also gives you an opportunity to present a critical appraisal of the project as a whole. This could include, for example, whether the methodology you have chosen and the programming language used were appropriate

Chapter 7

Conclusions and Future work

7.1 General Rules

The Conclusions section should be a summary of the aims of project and a restatement of its main results, i.e. what has been learnt and what it has achieved. An effective set of conclusions should not introduce new material. Instead it should briefly draw out, summarise, combine and reiterate the main points that have been made in the body of the project report and present opinions based on them. The Conclusions section marks the end of the project report proper. Be honest and objective in your conclusions.

7.2 Future Work

7.3 General Rules

It is quite likely that by the end of your project you will not have achieved all that you planned at the start; and in any case, your ideas will have grown during the course of the project beyond what you could hope to do within the available time. The Future Work section is for expressing your unrealised ideas. It is a way of recording that I have thought about this, and it is also a way of stating what you would like to have done if only you had not run out of time1. A good Future Work section should provide a starting point for someone else to continue the work which you have begun.

Appendix A

Collected Materials

Please be sure to put here any materials you have collected during your graduation project.

Bibliography

[1] M. Rehm, N. Bee, and E. Andre, "Wave like an egyptian: accelerometer based gesture recognition for culture specific interactions," in *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction - Volume 1*, ser. BCS-HCI '08. Swinton, UK, UK: British Computer Society, 2008, pp. 13–22.

[2] S. T. Ayman Atia and J. Tanaka, "Smart gesture sticker: Smart hand gestures profiles for daily objects interaction," *Computer and Information Science, ACIS International Conference on*, vol. 0, pp. 482–487, 2010.