King Saud University
College of Computer & Info Sciences
Department of Software Engineering

SWE 321 – SOFTWARE DESIGN AND ARCHITECTURES

Instructor: Dr. Zakarya Alzamil
Office: 2117, **Email:** zakarya@ksu.edu.sa

SEMESTER PROJECT

Goal and Description

The purpose of this project is to apply your software architectures knowledge and experience into a domain problem to produce a solution that can be considered as result of the knowledge and skills you gained. In this project you are required to propose a working solution for a selected real world domain problem. You should define the problem, specify, design, implement, and validate a working system. The architectural principle decisions made through the software development cycle should be identified. Also, any ideas, knowledge, patterns, engineering guidance, experience, structure, behaviors, implementations, or test suites that has been reused, should be highlighted in the development phases. Also, the learnt new ideas, knowledge, patterns, engineering guidance, or experience should be acknowledged.

Grading

1. Problem definition 10%
2. Architectural Document 30%
3. Design Document 30%
4. Demo, Source code, and final report 30%

Team Organization

Every student must work within a team. Each team consists of 3-4 students. There must be a leader or a coordinator for each deliverable (submission). Each team member must take a turn in the leadership, i.e., every team member must take a responsibility for one phase of the project development.

King Saud University
College of Computer & Info Sciences
Department of Software Engineering

GENERAL GUIDELINES FOR PROJECT DELIVERABLES

REQUIREMENT ANALYSIS

In this phase of the software life cycle you will play the role of a system architect. After interviewing the project sponsor/client you will clearly define the problem to be solved by the software system to be developed. In this phase you describe "what" the system will do, not "how" it will do it. However, design consideration should take place during requirement analysis at the macro level (system architecture), so the software architect should make his role by identifying the current/existing solution/experience in the domain problem, because the new solutions come from the observation of existing solution and their limitations. Also, the Non-Functional Properties should be raised in this phase. In this phase, the system architect involve with an interaction with the client/user to determine their needs. It also describes the system functions and these descriptions allow test team members to begin to design the final acceptance tests based on these functions that insure the customer's needs are met. The software engineering team must prepare for this phase by developing a list of questions designed to uncover the functional and qualitative requirements of the proposed software. They must arrange for an interview(s) with the client/user to get answers to these questions. Questions revolve around finding out some of the following:

1. *How the client currently solves the problem?*
2. *What the client expects the system to do (its functions)?*
3. *What are the performance requirements?*
4. *What is the problem domain vocabulary?*
5. *In what type of environment will the system run?*
6. *Will it interact with other software or hardware systems, etc.?*

When all these questions are answered and the team produces a document, the result will be a model of the system at a very high level of abstraction. The resulting document must then be presented to the client/user and evaluated by them. In order to prepare you for developing this document as a smoothly running team, the following activities are assigned. A description of items you need to include in the document appears in a next page.

Activities

1. *Prepare a list of questions for the client/user to determine functional requirements, as well as the non-functional properties.*
2. *Set up an interview(s) with the client and ask the questions.*
3. *Develop the problem definition.*
4. *Develop the document.*

King Saud University
College of Computer & Info Sciences
Department of Software Engineering

PROBLEM DEFINITION
(1st Deliverable)

The problem definition document for your project may be organized as follows:

1. *Cover page*: The cover page should contain the title of the project, a list of the author names (indicate the leader of the team), and (optionally) the contribution of each member of the team in the preparation of the document.

2. *Frontal matter*
    a. Table of contents.
    b. Revision Control History
        i. Version Number
        ii. Date
        iii. Description of change and the reason for change

3. *System Description & Purpose*: A description of the project, its purpose, and why it is useful and interesting to the team members.
4. *The System Context View:* Provide at least two user scenarios to describe how a user would interact with your system and what the benefit to them would be. These should include a scenario title and a description of the scenario including the inputs expected of the user and the outputs the system will provide to the user.
5. *Functions & NFP of the System:* A brief description of the functional and non-functional properties of your system.
6. *Challenges*: A discussion of any problems or challenges you expect to encounter.
7. *Projection:* There is a time limit for completion of this term project that is a reflection of real-world demands. In view of the time limit, and the expected performance, as well as the expected skill to be gained. Describe what you will accomplish by the end of the semester. This is done by creating an outline of functions of the system that you will finish within the period specified as well as the skills/knowledge you think you will gain.

King Saud University
College of Computer & Info Sciences
Department of Software Engineering

SYSTEM ARCHITECTURE DOCUMENT
(2nd Deliverable)

The architectural document for your project may be organized as follows:

1. *Cover page*: The cover page should contain the title of the project, a list of the author names (indicate the leader of the team), and (optionally) the contribution of each member of the team in the preparation of the document.

2. *Frontal matter*
    a. Table of contents.
    b. Revision Control History
        i. Version Number
        ii. Date
        iii. Description of change and the reason for change

3. *Introduction*: The document should start with an introductory paragraph describing the project and its goals. This section should also list of changes since the initial submission. Include those changes recommended by the instructor.

4. *Methodology:* How do you plan to implement your proposed architecture? Provide a plan that describes how the system will be designed and implemented (e.g., what modules/subsystems and in what order, who will do them and how they should be tested), with estimates for how long each module or subsystem will take. Provide a schedule for the remainder of the project with respect to these modules/subsystems, including the prototype demonstration, design deadline, final implementation, and final demonstrations. A Gantt chart is not required but may be helpful. Investigate and report on the feasibility of key or risky parts of your system.

5. *System Architecture*: Give the overall structure of the system that you have proposed with descriptions of each major component, the connectors, and the topology that binds them. List and describe the principle design decisions about the structure, behavior, interaction, and non-functional properties of the system you will build. Provide a set of views that describe the conceptual architecture (abstract level), including its structure, behavior, implementation, constraints, and so on. The components could be packages, subsystems, modules, frameworks, or external interfaces as is appropriate for your project. Concentrate on the goals, requirements, evolvability, interfaces, and testability of the entities in your architecture. Ensure that all of the external components/services your system will interact are accounted; these must be captured in your architectural description. The diagram should accurately reflect the system you want to build, be internally and externally consistent, and be as unambiguous as possible. Your architecture should be easy to understand with simple interfaces and modest interaction among components. Clarify the architectural style(s) of your system and justify why you

chose the style(s) you did. Include a discussion of at least another style you considered and justify why you decided not to use it in the end. While finding the right balance between architecture and design may be different, be careful not to make your architecture so high-level that it misses important information (e.g., if you have a client/server system, elaborate on the architectures of both the client and server components). A good rule of thumb is that discussing individual classes, methods, and fields is too low level. Your architectural document should contain some clarifying diagrams that clearly illustrate the structure of your system. The most important of these is the component diagram; be sure it is clear and provides interfaces for each component. You should use software tools to draw the architecture. Include a description (e.g., walk-through) of how each of the use cases you provided with your proposal activate or use the various parts of your system's architecture. Also describe how your architecture supports the non-functional properties you outlined in your proposal. Justify all architectural decisions. The structure of this subsection may be organized as follows:

a. Design decisions: list and describe the principle design decisions about the structure, behavior, interaction, and non-functional properties of your system.
b. Domain model: models the real-world system components whose behavior the system represents, they can be subsystems, packages, or modules. You should identify any existing architecture, experience, ideas, and/or components that might be of any benefit for this system under development.
c. Architectural Style: describe the architectural style(s) of your system and justify why you chose the style(s) you did. Include a discussion of at least another style you considered and justify why you decided not to use it in the end.
d. Structural model: determines the components and connections between them. Also include DBs, data files, 3rd party components. Focus on the high level; do not focus on classes, variables, etc. Also, focus on interactions between components, not on interactions within them. Explain the functionality of each component. Explain how your architecture meets your non-functional properties. Relate architecture to the styles that you have learned. Mention another possible style & discuss it. Do not over complicate your design. System should be easy to understand with simple interfaces between components. The structural model may be build via the following steps:
    i. Identify the components and develop the initial diagram based on the style/model you choose.
    ii. Identify any DBs, data files, and 3rd party components in general.
    iii. Analyze the behavior of the system via developing the use-cases at macro level (system-subsystem) that map on the execution and implementation architectures.
    iv. Develop the walk-through for each use case.

e. User interface: In this section, you should describe the user interface of your system with the consideration of the principles of good interface design: visibility, affordance and usability, and feedback. As well as considering the cognitive psychology, background and knowledge of the users, sociology, and linguistics. Provide sample screens of the system supporting major features of the system with short description

6. *Non-Functional Properties*: Analyzes the quality attributes by identifying four quality-attribute-related needs and the corresponding architectural responses. Write a paragraph on each, explaining why they are significant to the system. Also, list and describe any other constraints such as hardware, software, timing, size, reliability, etc.

7. *Quality Assurance*: How will you ensure that your solution is acceptable? Specify the kinds of tests and reviews that you are using to ensure the quality of the delivery. Do not include specific test cases, only general test types. Also, describe how the client will determine the quality of the system. Specific items to address:
    a. Reviews. What type and when will the development team conduct reviews.
    b. Verification. How will you assure that the developers built what was specified in the design? How will your assure that the developers followed standards and regulation.
    c. Validation. How will you establish confidence in the client that: 1) requirements are met. 2) the software is secure, 3) it is reusable and expandable, 4) performance, etc?
    d. Acceptance Criteria. How does the user install the software? What training and technical documentation is provided?
        i. Procedures
        ii. Testing
        iii. Training
        iv. Documentation

8. *Future Considerations:* Can your system accommodate future changes? What are possible future changes/additions to your system? How does your architecture support these changes? Is there anything that must be changed in your architecture? What are the key parts of your system? Are their parts that are more risky than others? How are you going to check the feasibility of these parts?

King Saud University
College of Computer & Info Sciences
Department of Software Engineering

SYSTEM DESIGN DOCUMENT
(3rd Deliverable)

This document describes, "how the system will do it." The software design consists of two parts: the architectural design and the detailed design. The architectural design has been described in the previous deliverable "*system architecture document*". The detailed design is a refinement of the architectural design in which more elaboration is performed on the components and connectors, such that more details on behaviors, data, interfaces, dependencies, etc are determined with concentration on dynamic behaviors of the system.

The design document may be organized as the following:

1. *Cover page*: The cover page should contain the title of the project, a list of the author names (indicate the leader of the team), and (optionally) the contribution of each member of the team in the preparation of the document.

2. *Frontal matter*:
    a. Table of contents.
    b. Revision Control History
        i. Version Number
        ii. Date.
        iii. Description of change and the reason for change.

3. *Introduction*: Describe the purpose and scope of the specification document.

4. *Detailed design:* You should provide a clear specification for each component, so that the developer implements it. The detailed design will include a clear description of the behavior of the component and its externally visible interfaces. It will include a description of algorithms and data structures to be used and will describe non-obvious implementation techniques. Rationale must be provided documenting why you selected your design. The applicability of your design compared to alternative designs could also be referenced in this discussion. You should reference descriptions of important patterns, component abstractions, data structures or algorithms that are critical to the successful implementation of your system. You should also include a state diagram and sequence diagrams for each of the use cases you specified in your specification document. Use diagrams as appropriate for this report. At a minimum, include a components diagram that shows all of the component and public API for your system and how they interact. Clarify the physical location of where the components will reside (e.g., in the browser, on a server), as well as any external API your system will use. External Interfaces: Give details of information transmitted to/from the system, such as the Graphical User Interfaces, files, databases, messages or networks. This should include menu design (if appropriate) and a clear specification of any details by which components in the system transmit information to/from the external interfaces. This detailed design should be a companion to your previously-

submitted system architecture. Ideally, your report should repeat as little of the material as possible from the system architecture while still remaining a standalone document as much as possible. The structure of this subsection may be organized as follows:

a. Detailed system architecture: refine the system architecture developed in the previous phase, to elaborate on the component diagram, DBs, system behavior etc, and to represent the inheritance structure, such that a specification for each component is provided as follows:
    i. Component Name
    ii. Description
    iii. Properties/data
    iv. Behavior/functionality
    v. Connectors and Interfaces
    vi. Dependencies
    vii. Resources
b. Dynamic model: refine the system behavior to describe the dynamic behavior of the system by depicting the state and sequence diagrams.
c. Implementation consideration: describe the mapping of the system architecture into an implementation artifact, in which the architecture intent should be reflected in the constructed system. Describe the mapping of the component, connectors, interfaces, configuration, and NFP, etc. describe the used framework, middleware, and non-obvious implementation techniques. Also, you should provide a description of important algorithms, patterns, component abstractions, data structures or algorithms that are critical to the successful implementation of your system. Show all of the component and public API for your system and how they interact. Give details of information transmitted to/from the system, such as the Graphical User Interfaces, files, databases, messages or networks.

PRESENTATION and FINAL REPORT
(4th Deliverable)

PRODUCT DEMO

You are asked to present your product to a committee that consists of the course instructor along with the teaching assistant(s). Your presentation should outline your work in this project from the problem definition till the final product. You should present your product in formal way and be ready to answer any question regarding your project. You will be given a hint of the evaluation criteria for your presentation at the end of the semester.

FINAL REPORT

The final report should be a companion to your previously-submitted system architecture and design. It may be organized as the following:

1. *Cover page*: The cover page should contain the title of the project, a list of the author names (indicate the leader of the team for the implementation and testing phase), and (optionally) the contribution of each member of the team.

2. *The architecture and design documents*: Be sure that any changes are recorded in those documents. With all changes noted. Make sure you omit the cover page as well as the frontal matter page.

3. *Implementation Differences:* You should describe the differences between your proposed architecture and design with your final product. The final document should capture the differences and (most importantly) concretely explain the rationale for these differences. Discussing your architecture in terms of its conceptual notion vs. concrete realization may make sense for your project. This may help the developers understand how and why your architecture and design evolved from their proposed versions to the final form that exists in the source code.

4. *Architecture Analysis (testing the architecture):* You should perform the four "C" analyses to ensure that the system architecture is complete, consistent, compatible, and correct. Describe the method and process used for such analysis. Also, list the analyzed system components (provide examples of test cases), and state any known problems (bugs) in the system architecture.

5. *Deployment and Mobility*: describe the process of placement of a system's software components on its hardware hosts including its resources needed by the system's components (hardware, network, peripheral devices, system software, other application software, and data resources). Also, describe any changes of the deployment of a component during runtime.

6. *User manual*: Provide a short user manual for the system. The first section should be the installation procedures. Be sure to test this procedure and include the passwords necessary to install the system.

7. *Personal Reflection:* referring to the challenges and projection you expected in the 1st deliverable, reflect on how this project challenged your architectural, design, and development skills. Do you feel more comfortable architecting systems now than you did before? Do you feel like a more competent system designer? By implementing your design, did you learn new skills that will help you design more effectively in the future? What aspects of the course helped you improve the quality of your architecture and design and what aspects hindered you? How would you modify this course to help future software engineers become more competent architects and designers?