Department of
Computer Science and Engineering

Title: Develop System Architecture (Part I: High Level Design) for a specific project

Integrated Design Project II

CSE 406

Green University of Bangladesh

1 Objective(s)

- To define a comprehensive solution based on principles, concepts, and properties logically related to and consistent with each other of a system.
- To design high level architecture for the given project.
- To understand the motive of high level design for a project.

1.1 Sub-Objective(s)

- To describe stakeholders' communication.
- To describe the functionalities performed by the system.
- To reuse the same components for systems with similar requirements in large scale.
- To represent the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them.

2 Problem analysis

Architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system. In the model of the software development process, architectural design is the first stage in the software design process. It is the critical link between design and requirements engineering, as it identifies the main structural components in a system and the relationships between them. The output of the architectural design process is an architectural model that describes how the system is organized as a set of communicating components.

In practice, there is a significant overlap between the processes of requirements engineering and architectural design. Ideally, a system specification should not include any design information. This is unrealistic except for very small systems. Architectural decomposition is usually necessary to structure and organize the specification. Therefore, as part of the requirements engineering process, you might propose an abstract system architecture where you associate groups of system functions or features with large-scale components or sub-systems. You can then use this decomposition to discuss the requirements and features of the system with stakeholders.

Architectural design is a creative process where you design a system organization that will satisfy the functional and non-functional requirements of a system. Because it is a creative process, the activities within the process depend on the type of system being developed, the background and experience of the system architect, and the specific requirements for the system. It is therefore useful to think of architectural design as a series of decisions to be made rather than a sequence of activities.

During the architectural design process, system architects have to make a number of structural decisions that profoundly affect the system and its development process. Based on their knowledge and experience, they have to consider the following fundamental questions about the system:

- How will the system be distributed across a number of cores or processors?
- What architectural patterns or styles might be used?
- What will be the fundamental approach used to structure the system?
- How will the structural components in the system be decomposed into subcomponents?
- What strategy will be used to control the operation of the components in the system?
- What architectural organization is best for delivering the non-functional requirements of the system?
- How will the architectural design be evaluated?
- How should the architecture of the system be documented?

Problem statement and motivation for the given project: The project entitled Banking ATM system has a drastic change to that of the older version of banking system, customer feel inconvenient with the transaction method as it was in the hands of the bank employees. In our ATM system, the above problem is overcome here, the transactions are done in person by the customer thus makes the customers feel safe and

secure.Thus the application of our system helps the customer in withdrawing money, checking the balance and transaction of the amount with mini-statement and transferring the balance by validating the pin number therefore ATM system is more user friendly.

3 Methodology

You can design software architectures at two levels of abstraction, which we call architecture in the small and architecture in the large:

1. Architecture in the small is concerned with the architecture of individual programs. At this level, we are concerned with the way that an individual program is decomposed into components. This chapter is mostly concerned with program architectures.

2. Architecture in the large is concerned with the architecture of complex enterprise systems that include other systems, programs, and program components. These enterprise systems are distributed over different computers, which may be owned and managed by different companies.

The apparent contradictions between practice and architectural theory arise because there are two ways in which an architectural model of a program is used:

1. High Level Design: As a way of facilitating discussion about the system design A high-level architectural view of a system is useful for communication with system stakeholders and project planning because it is not cluttered with detail. Stakeholders can relate to it and understand an abstract view of the system. They can then discuss the system as a whole without being confused by detail. The architectural model identifies the key components that are to be developed so managers can start assigning people to plan the development of these systems.

2. Low Level Design (Detailed Design): As a way of documenting an architecture that has been designed The aim here is to produce a complete system model that shows the different components in a system, their interfaces, and their connections. The argument for this is that such a detailed architectural description makes it easier to understand and evolve the system.

3.1 Architectural views

It is impossible to represent all relevant information about a system's architecture in a single architectural model, as each model only shows one view or perspective of the system. It might show how a system is decomposed into modules, how the run-time processes interact, or the different ways in which system components are distributed across a network. All of these are useful at different times so, for both design and documentation, you usually need to present multiple views of the software architecture.

1. A logical view, which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.

2. A process view, which shows how, at run-time, the system is composed of interacting processes. This view is useful for making judgments about nonfunctional system characteristics such as performance and availability.

3. A development view, which shows how the software is decomposed for development, that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful for software managers and programmers.

4. A physical view, which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.

3.2 Architectural Patterns

The idea of patterns as a way of presenting, sharing, and reusing knowledge about software systems is now widely used. Different types of architectural patterns are shown in Fig.1, Fig.2, Fig.3, Fig.4, Fig.5, Fig.6 and Fig.7.

3.3 Application Architecture

Application systems are intended to meet a business or organizational need. All businesses have much in common they need to hire people, issue invoices, keep accounts, and so on. Application architectures encapsulate the principal characteristics of a class of systems. For example, in real-time systems, there might be generic architectural models of different system types, such as data collection systems or monitoring systems. Although instances of these systems differ in detail, the common architectural structure can be reused when developing

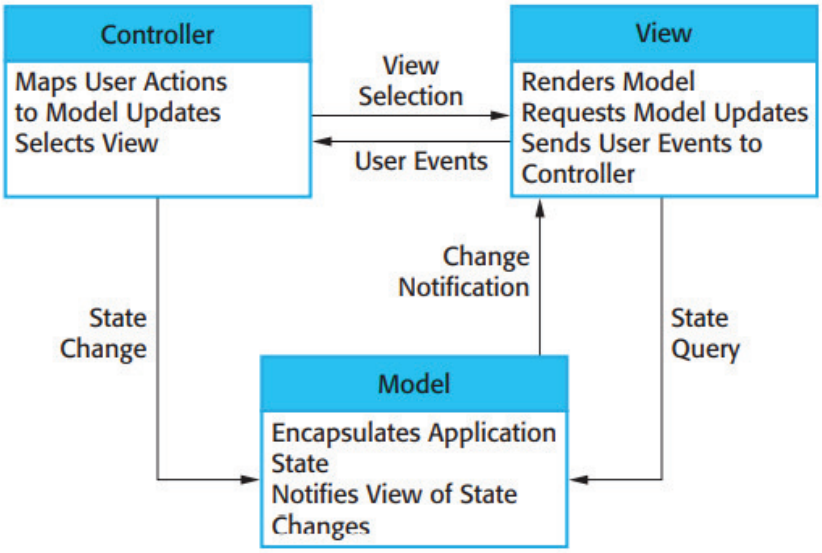

Figure 1: Model-View-Controller pattern

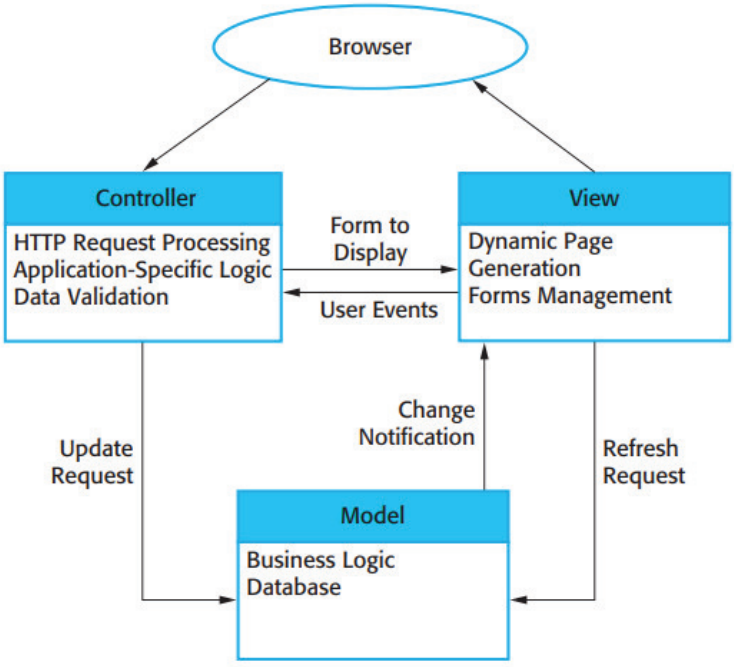

Figure 2: Web based Model-View-Controller pattern

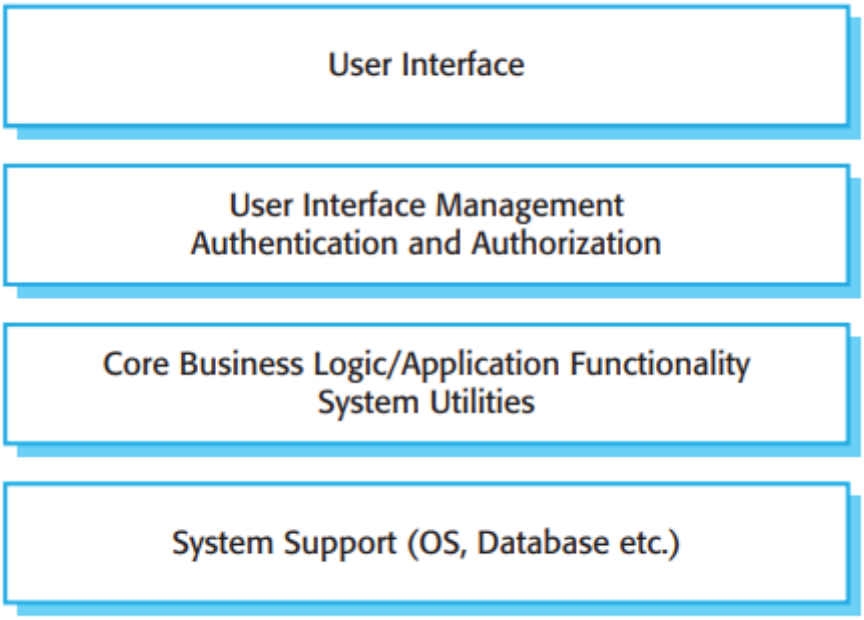

Figure 3: The layered architecture pattern

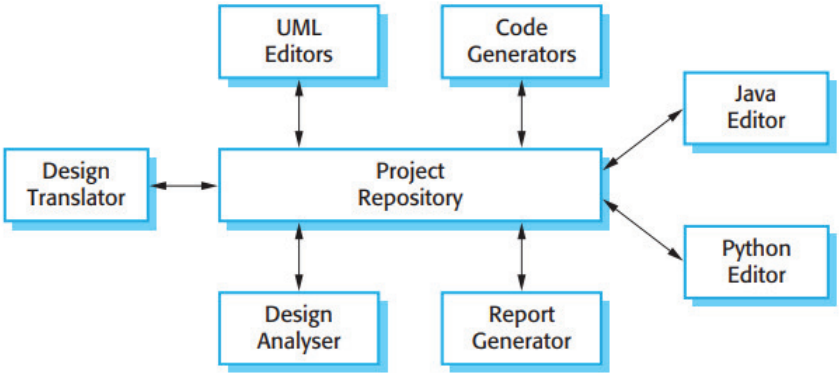

Figure 4: A repository architecture for an IDE

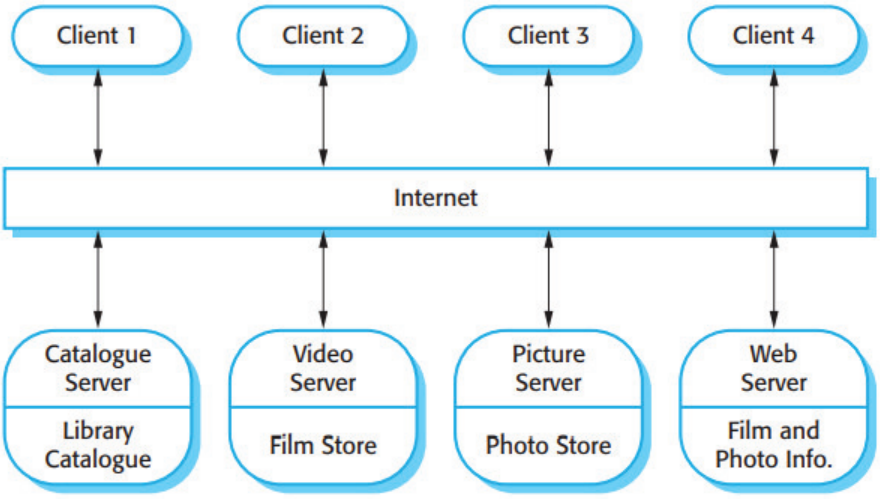

Figure 5: A client—server architecture pattern for a film library

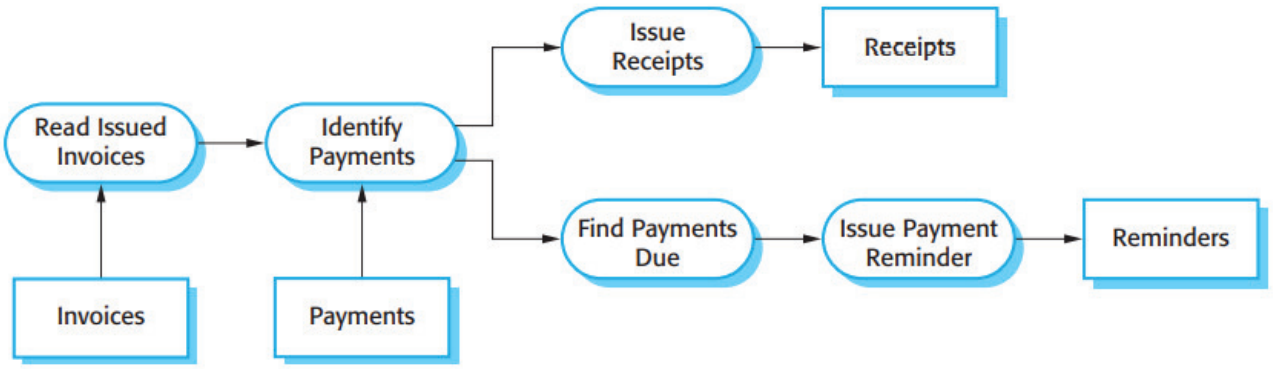

Figure 6: An example of the pipe and filter architecture

new systems of the same type. The application architecture may be re-implemented when developing new systems but, for many business systems, application reuse is possible without re-implementation.

A generic system is configured and adapted to create a specific business application. As a software designer, you can use models of application architectures in a number of ways:

- As a starting point for the architectural design process If you are unfamiliar with the type of application that you are developing, you can base your initial design on a generic application architecture. Of course, this will have to be specialized for the specific system being developed, but it is a good starting point for design.
- As a design checklist If you have developed an architectural design for an application system, you can compare this with the generic application architecture. You can check that your design is consistent with the generic architecture.
- As a way of organizing the work of the development team The application architectures identify stable structural features of the system architectures and in many cases, it is possible to develop these in parallel. You can assign work to group members to implement different components within the architecture.
- As a means of assessing components for reuse If you have components you might be able to reuse, you can compare these with the generic structures to see whether there are comparable components in the application architecture.
- As a vocabulary for talking about types of applications If you are discussing a specific application or trying to compare applications of the same types, then you can use the concepts identified in the generic architecture to talk about the applications.

For example as a transaction processing applications: Transaction processing applications are database-centered applications that process user requests for information and update the information in a database. These are the most common type of interactive business systems. They are organized in such a way that user actions can't interfere with each other and the integrity of the database is maintained. Fig.**??** shows the example of transaction processing application architecture.

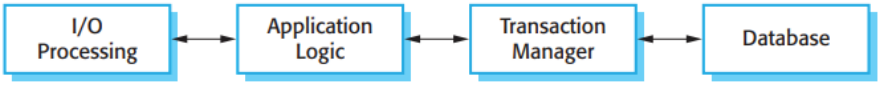

Figure 7: Example of a Transaction processing application architecture

3.4 C4 Model for System Architecture

C4 model (as shown in Fig.8, Fig.9 and Fig.10) provides a way for software development teams to efficiently and effectively communicate their software architecture, at different levels of detail, telling different stories to different types of audience, when doing up front design or retrospectively documenting an existing codebase. Detailed description of the C4 model will get from https://c4model.com/.

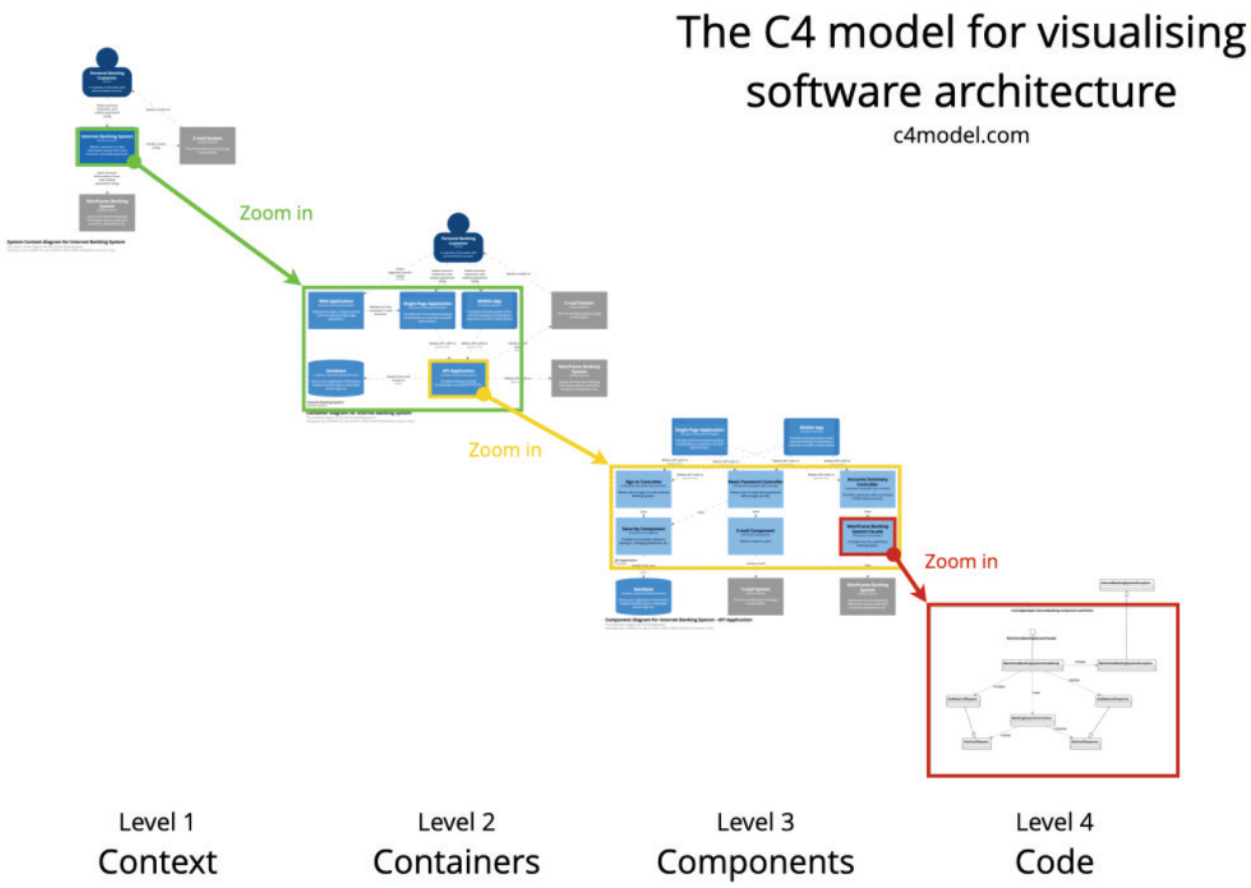

Figure 8: C4 Model for system architecture

3.5 Required Software

There are several software available that can be used online and offline to draw these system architectures like as follows.

1. Visual Paradigm for UML 8.2 (online link: https://online.visual-paradigm.com/)
2. StartUML
3. Lucidchart and other drawing tools

4 Implementation

4.1 High Level Design of The Banking ATM System

4.1.1 High Level Design of The Banking ATM System Based on Transaction Processing Application Architecture

High Level Design in short HLD is the general system design means it refers to the overall system design. It describes the overall description/architecture of the application. It includes the description of system architecture, data base design, brief description on systems, services, platforms and relationship among modules. It is also known as macro level/system design. It is created by solution architect. It converts the Business/client requirement into High Level Solution. It is created first means before Low Level Design. The high level design of system architecture of the Banking ATM system based on Transaction Processing Application architecture is shown in Fig.11.

4.1.2 High Level Design of The Banking ATM System Using Abstract View of Context

The Fig.12 shows the High level architecture (abstract view) of the system of A Banking ATM based on context based architecture.

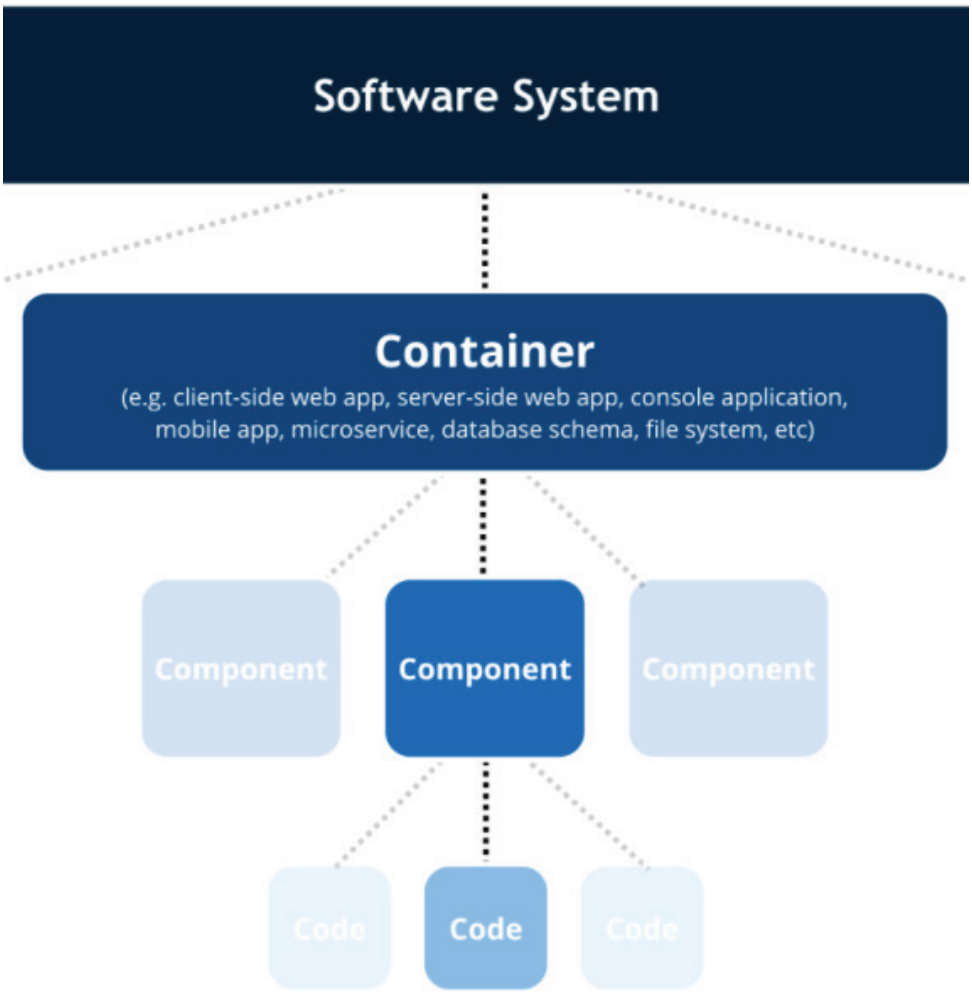

Figure 9: Details of the C4 Model for system architecture

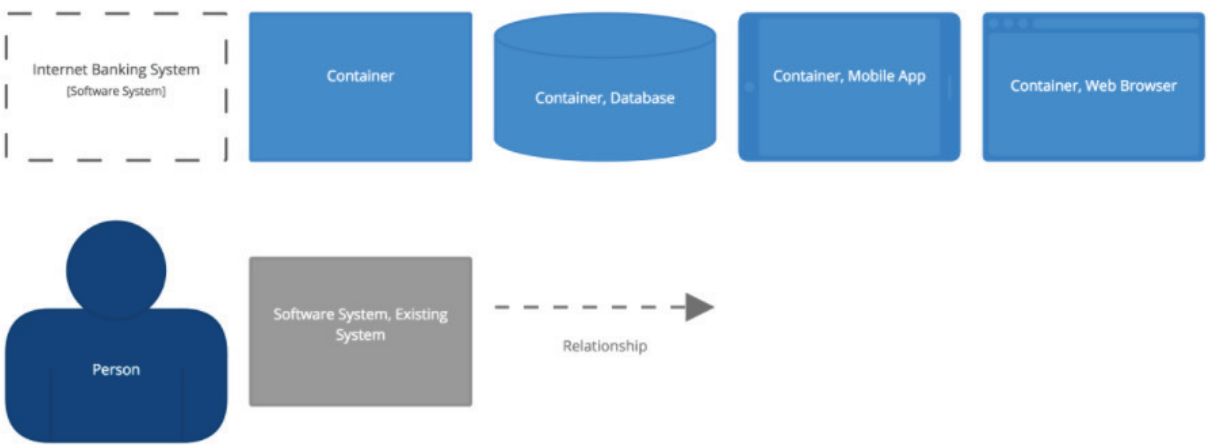

Figure 10: Basic notations of the C4 Model for system architecture

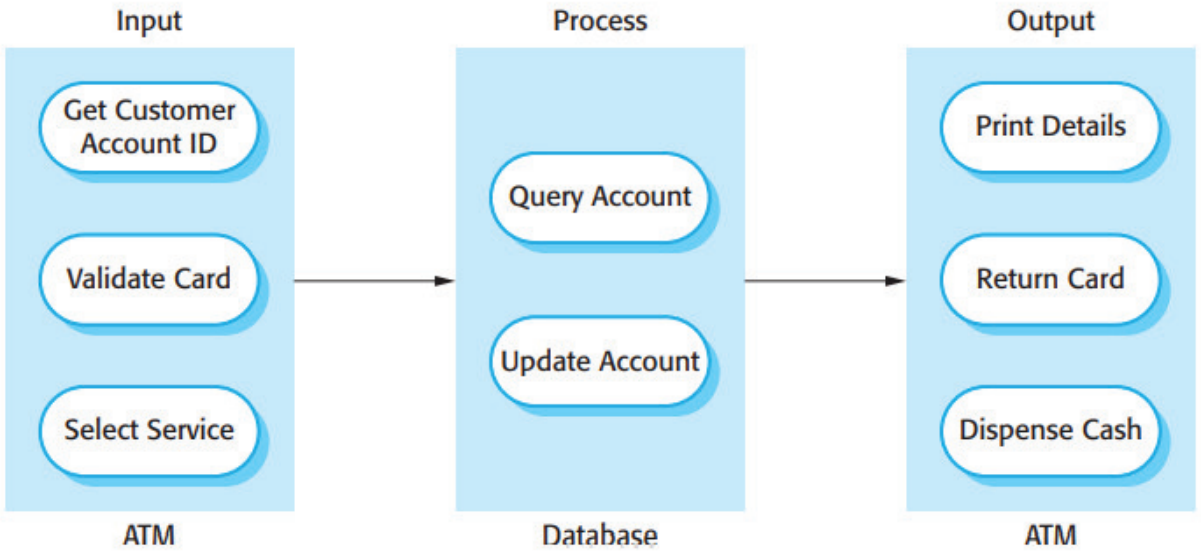

Figure 11: High level view of the system architecture A Banking ATM system based on transaction processing application architecture

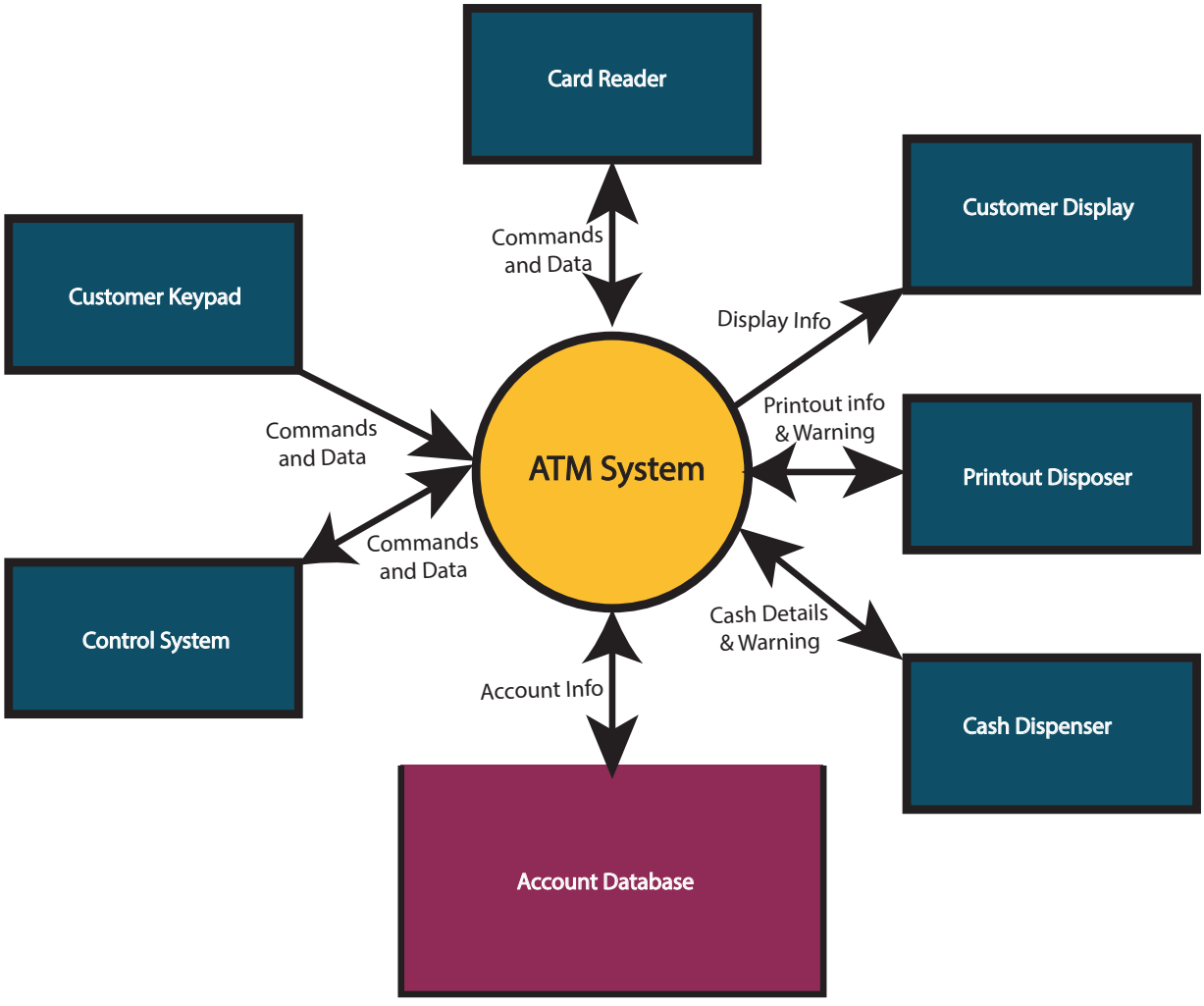

Figure 12: High level architecture (abstract view) of the system of A Banking ATM based on context based architecture

You can go through the following link to find the detailed explanation of the high level and low level architecture of the Banking ATM system.
https://link.springer.com/content/pdf/bbm%3A978-3-642-56209-9%2F1.pdf

5 Discussion & Conclusion

Based on the focused objective(s) to understand about high level of the system architectures including different patterns and views, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

6 Lab Task (Please implement yourself and show the output to the instructor)

1. Draw a high level architecture (abstract view) of the given project 'Library Management System'.

6.1 Problem analysis

A library database system is an infrastructure that allows users to search books and book content, add/remove, and download selected books. The problem faced is that library users require an efficient method to find a specific book or keyword(s) within a book given a continuously expanding library. Some scenarios of the 'Library Management System' are as follows:

1. User who registers himself as a new user initially is regarded as staff or student for the library system.

(i) For the user to get registered as a new user, registration forms are available that is needed to be fulfilled by the user.

(ii) After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to cardholder or user.

2. After getting the library card, a new book is requested by the user as per there requirement.
3. After, requesting, the desired book or the requested book is reserved by the user that means no other user can request for that book.
4. Now, the user can renew a book that means the user can get a new due date for the desired book if the user has renewed them.
5. If the user somehow forgets to return the book before the due date, then the user pays fine. Or if the user forgets to renew the book till the due date, then the book will be overdue and the user pays fine.
6. User can fill the feedback form available if they want to.
7. Librarian has a key role in this system. Librarian adds the records in the library database about each student or user every time issuing the book or returning the book, or paying fine.
8. Librarian also deletes the record of a particular student if the student leaves the college or passed out from the college. If the book no longer exists in the library, then the record of the particular book is also deleted.
9. Updating database is the important role of Librarian.

7 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.