Psychological Web Application

Requirements Specification For Intelligent psychological System

Eng./ Mahmoud Mouhamed Ahmed Taha
Eng./ Yasser Emad Ibrahim
Eng./ Aya Salah Abdel-Salam
Eng./ Mahmoud El-Diin Tarek

System Analysis version two desiccation for spring Term 2017-2018

Computer Science Department
University

Contents

Chapter 1

INTRODUCTION

1.1 MOTIVATION

The psychology is the science, It aims to understand the behavior of others and gather information about the way the brain works in order to better serve humanity, Psychology has four main goals:

1. **Describe behavior:**
   first goal of psychology is to describe behavior. description involves naming and classifying behavior. This description is based on careful, systematic procedure in contrast to haphazard description of common sense. Description is very important in that it makes you clear about what the phenomena under study. only after we described the behavior or phenomenon clearly we can move to the other goals.

2. **Understand or explain behavior:**
   The second goal of psychology becomes explaining the behavior or phenomenon that was described. psychologists who are concern this goal try to find out why such behavior occur. they take help of existing theories and knowledge to explain or understand behavior. in some cases if there are no theorizes or researches that can explain such behavior psychologists make tentative statements and try to test such hypothesis.

3. **Predict the behavior:**
   Another important goal for psychologists is to forecast future event. By carefully analyzing the relationship between different variables, psychologists can accurately predict what will be the relation in future between them. prediction helps in modifying the behavior. it is facilitated by understanding of the relationship.

4. **Control or modify behaviors:**
   The fourth goal of psychology is to control, modify or change the existing behavior. the behaviors that need to be corrected are modified through the help of psychological techniques. Only psychologists who work in applied are of psychology are concerned with controlling the behaviors.

A psychologist interested in gender and women's issues teaches at the communty college and workes with her college and community to elimi-

nate sexual harassement, and the researcher works at Educational Testing service to ferret out possible cultural bias in psychological tests. ***Contemporary Approaches to psychology***

- **The Behavioral Approach:**
  stresses that behavior is determined not only by environmental conditions but also by how thoughts modify the impact of environment on behavior.
- **Psychosomatic approach:**
  Emphasizes the unconscious aspects of the mind, conflict between biological instincts and society's demands and early family experiences.
- **Cognitive Approach:**
  focuses on the mental processes involved in knowing how we direct our attention, perceive, remember, think, and solve problems.
- **Behavioral neuroscience Approach:**
  views understanding the brain and nervous system as central to understanding behavior, thought, and emotion.
- **Evaloutionary Psychology Approach:**
  Emphasizes the importance of functional purpose and adaptation in explaining why behaviors are formed, are modified, and survive.
- **Sociocultural Approach:**
  Emphasizes social and cultural influences on behavior.

In this chapter we will introduce an overview at all flowing chapters and look at the purpose of documentation , then introduce the scope of document, and introduce some definitions' terms and acronyms that will use in the document, then mention the References that import from it some information that serve this document, finally we will Describe in generally the contents of this document

1.2 PURPOSE

main purpose from this document build Online System to manage daily work on the psychological center and illustrate and explain The system Of psychological center and all contents that included in it, and introduce or explore the problems and statues in the psychological center, and knowing the constraints, and introduce some efficient solutions that solve the specific problem, and representing the functional and nonfunctional of system.

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

we will introduce in

- **chapter one**
  in this chapter will introduce general description of the project and product functions and constraints
- **chapter two**
  will introduce the functional, performance, interface and explain the database
- **chapter three**
  will introduce the results and the features of the system
- **chapter four**
  will introduce the Appendices
- **chapter five**
  will introduce the index

This Project is a prototype for intelligent Management system and it restricted to create the daily tables of appointments of sessions and courses of the training department, send official confirmation message to clients and psychologists to confirm the reservation of the flow session, compute the total income and total outcome, then calculate the revenue in gaily, monthly, annually, calculate the salaries of employees and output ratio of psychologists then outcome the total amount per day, and finally generate some reports to help head Managers to make decisions in the fast time, we will introduce this in the flowing sections in details

1.4 PROJECT SCOPE

scope of this document is helping the employees in the psychological center and improve the methods that them was used to it, and helping the psychologists to work with more efficient and saving the time of them, and helping the head managers to take the correct decisions in the appropriate time, we will use some devices and techniques like fingerprint device to manage the attendance on the center

Chapter 2

Overall description

2.1 Product perspective

The product is supposed to be an open source, under the GNU general Public License. It is a web based system implementing client-server model. The Software Requirements Specification (SRS) For Inelegant psychological System provides simple mechanism for users to share and acquire knowledge.

This product have Database (DB) that store

2.1.1 Database System:

- **Session details:**
  Will record the information of any session like duration of the session , date , the client that take this session, the psychologist that supervise in this session , and report of this session.

- **Client description:**
  It includes client code, name address, email, phone number and birth-date and maybe other information, this information used for confirmed reservation of session and keeping the records of the client for any emergency or any other kind of information.

- **Specialist description:**
  It includes specialist code, name, address, and email, phone number, Social Security Number (SSN) , his qualifications, and graduation year, this information to performance evaluation and develop him.

- **Reservation description:**
  It includes the day of reservation, date, time, client details, and specialist details, this information to save right of client and specialist in the center.

- **Pay description:**
  It includes pay code, receptionist name, date, time, and price of session, thin total.

- **Income description:**
  It includes the client details, specialist details, session details, number of hour, and notes, this information to compute the total income per day, then per month, then per year and output the report.

- **exported description:**
  It includes The specialist details, total number of sessions, date, day, and total Proportion of Specialist, and total number of hours, this information to compute the total exported at month, and year.

- **report description:**
  It includes the head title, body of report, and footer and signature

2.1.2 client side:

The system deviated two subsystems, the first subsystem is client side services, and the second server side services, we will introduce in this subsection the client side services.

We provide in the client side services the main services like the user will able to log-in into system and log-out from it, he can send request of reservation to specific appointment, and retrieve his schedule, he can access his report that specified to his case, the user can update his profile and change his information, the specialist can enter the available appointment that appropriate to him, and retrieve his schedule, upload case's reports, and can connect with his cases and receptionist or management.

The client side services provide to management some reports for calculations of the center, report for total hours of all specialists, and total work hours of center, report for Exports of the center like total salaries of all Employees in the center, Total cost of all maintenance of all things monthly or yearly, report to help decision makers for take the correct decision, report for all problems in the center, and provide for all managers to connect with Employees and clients of center, the client side provide to center mechanism to confirm automatically or manually reservations of the next day.

2.1.3 Server side:

We provide in the server side services the main services like Encrypt all data to provide security to system, controller functions that access and connect with Database Management System (DBMS) that is a collection of programs that enables users to create and maintain a database, and provide the functions that can access or send and receive the requests from client side services

2.2 PRODUCT FEATURES

The major features of Inelegant psychological database system as shown in below entity relationship model

2.3 User characteristics

2.3.1 Receptionist

Receptionist should be able to determining the specific appointment for the client, and update any appointment according to client, he should able to retrieve all information of any reservation appointment, retrieve any information of calculation system to specific client, and he retrieve all his conversations with the client in the system

2.3.2 Specialist

Specialist should be able to enter his appointments of his sessions daily, or monthly, and he retrieve his schedule of day, he can update his appointments before passing 24 hours from setting it,should be able to retrieve all information of his cases and specific reports that it was written by him, and able to upload the report of specific case.

2.3.3 Client

Client should able to retrieve his schedule of his appointments, able to choose from available appointments that his specialist determined its, and able to retrieve all reports that specific his case and download this report, able to update his information and update the appointment before passing 24 hours from his choosing it, he can see the program that his specialist preparing it to him and Proposed Plans to solve issues of the case, he can connect with management of center.

2.3.4 Trainee

Trainee should able to choose from available courses and Enroll it, able to retrieve his schedule of appointments of all courses that he Enroll its and all information about its, able to access all materials that specified to the course and download its, able to retrieve specific report about Trainer, his Performance, his Tests, and results or points that take from Exercises

2.3.5 Trainer

Trainer should be able to set all courses that will Lecture its, and determine the appropriate appointments for him, supervise trainee that enrolled to courses that will be specified, he can upload the materials of the course and retrieve all information that specific of the trainee in the course, and dropped out The rioters from the course, connect with trainees and receptionist

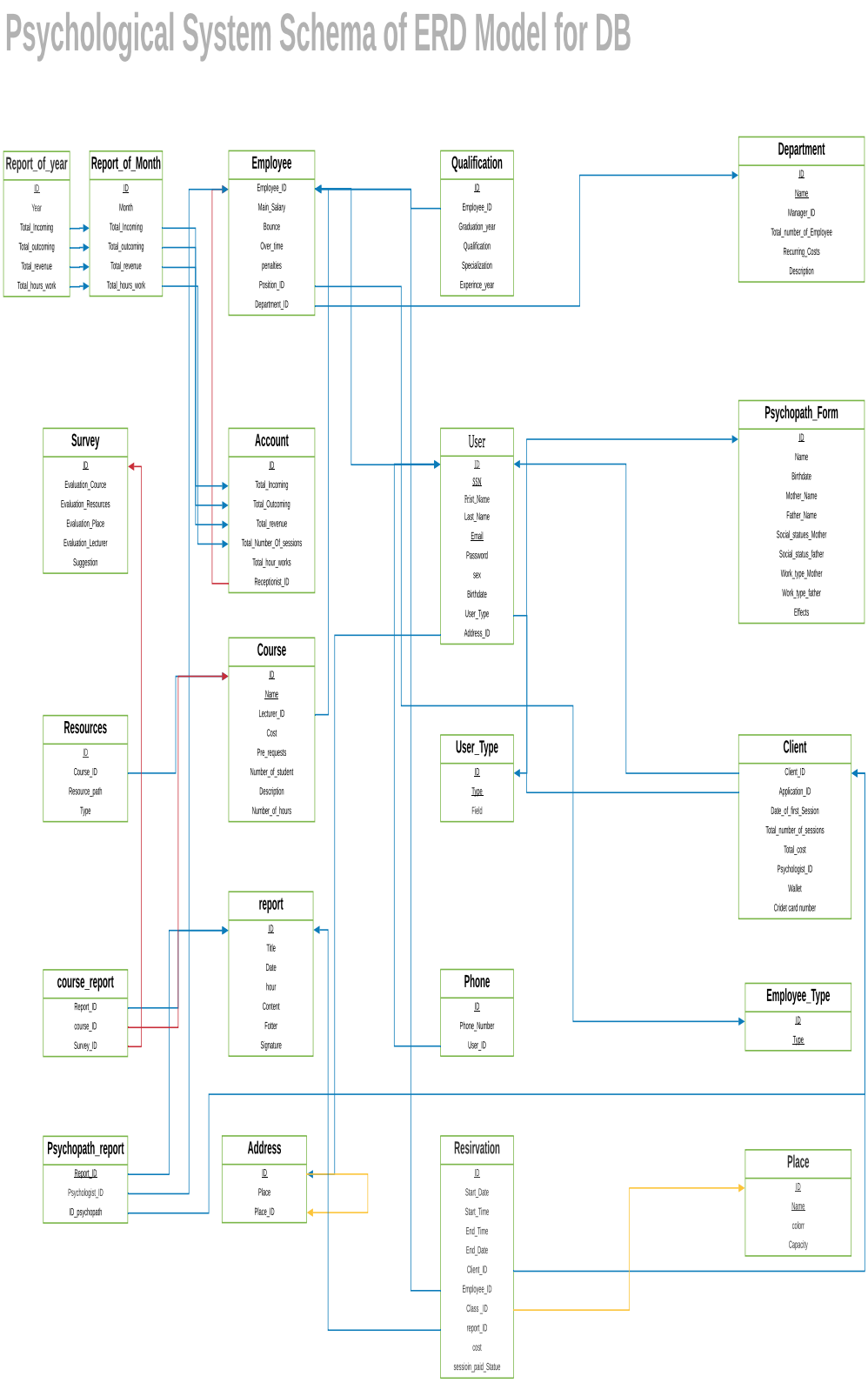
2.4 OPERATING ENVIRONMENT

Operating environment for Inelegant psychological System is as listed below.

- **distributed database**
- **client/server system**
- **Operating system: Windows, windows phone, android, mac os, linux**
- **database: my sql database**
- **platform: Python/Java/Java Server Pages (JSP)**

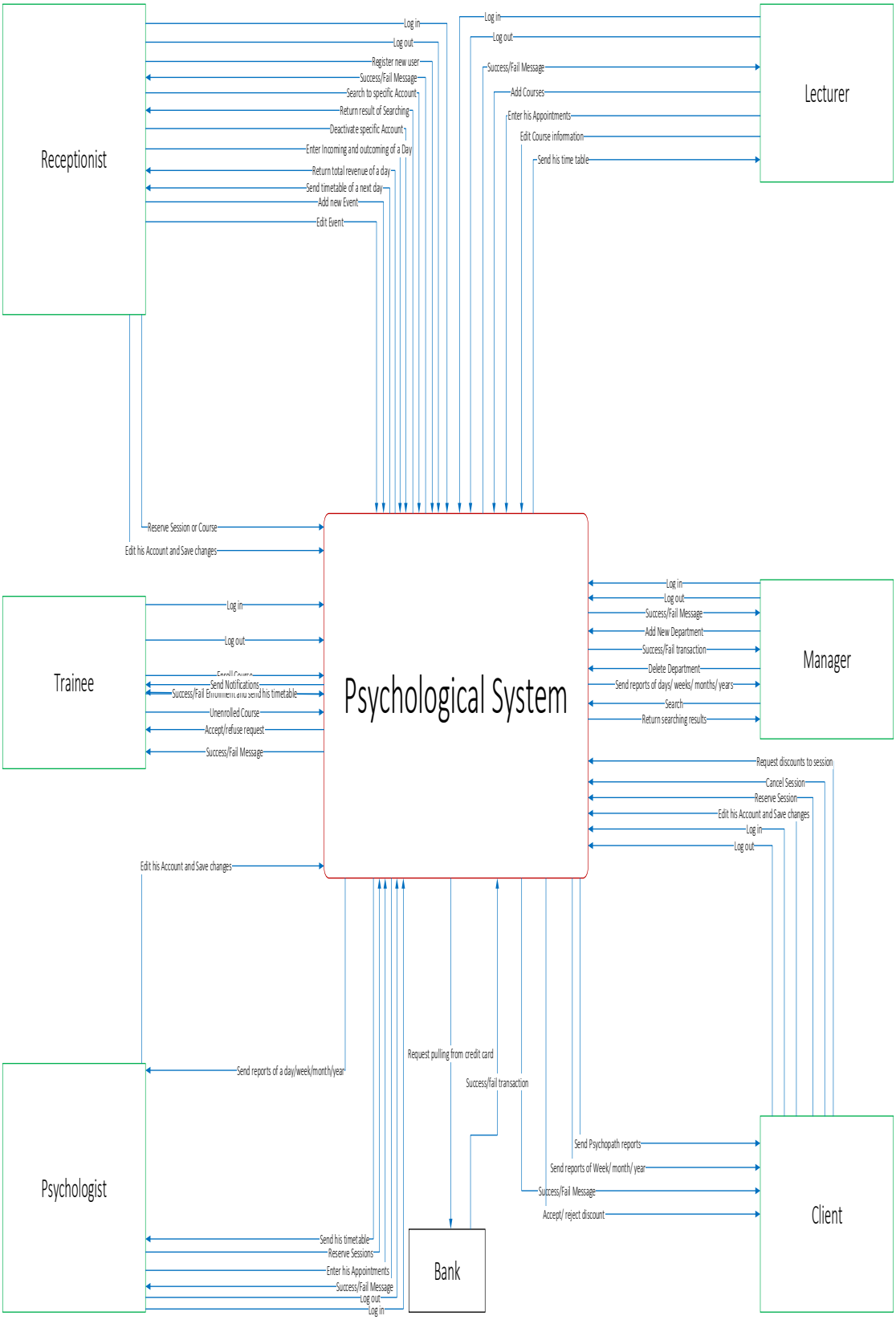
2.5 DESIGN AND IMPLEMENTATION CONSTRAINTS

2.5.1 Schema

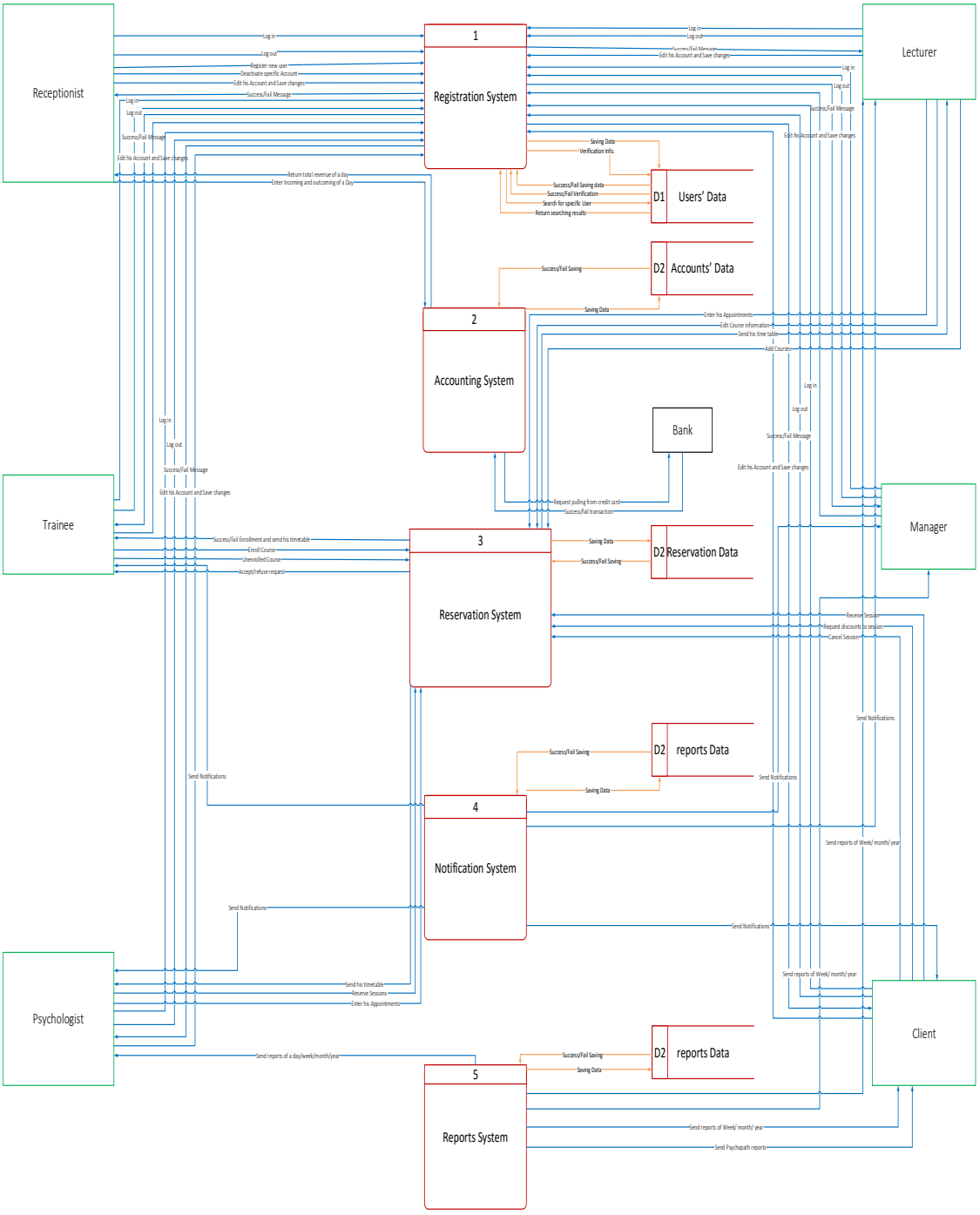


2.5.2 Data flow diagram

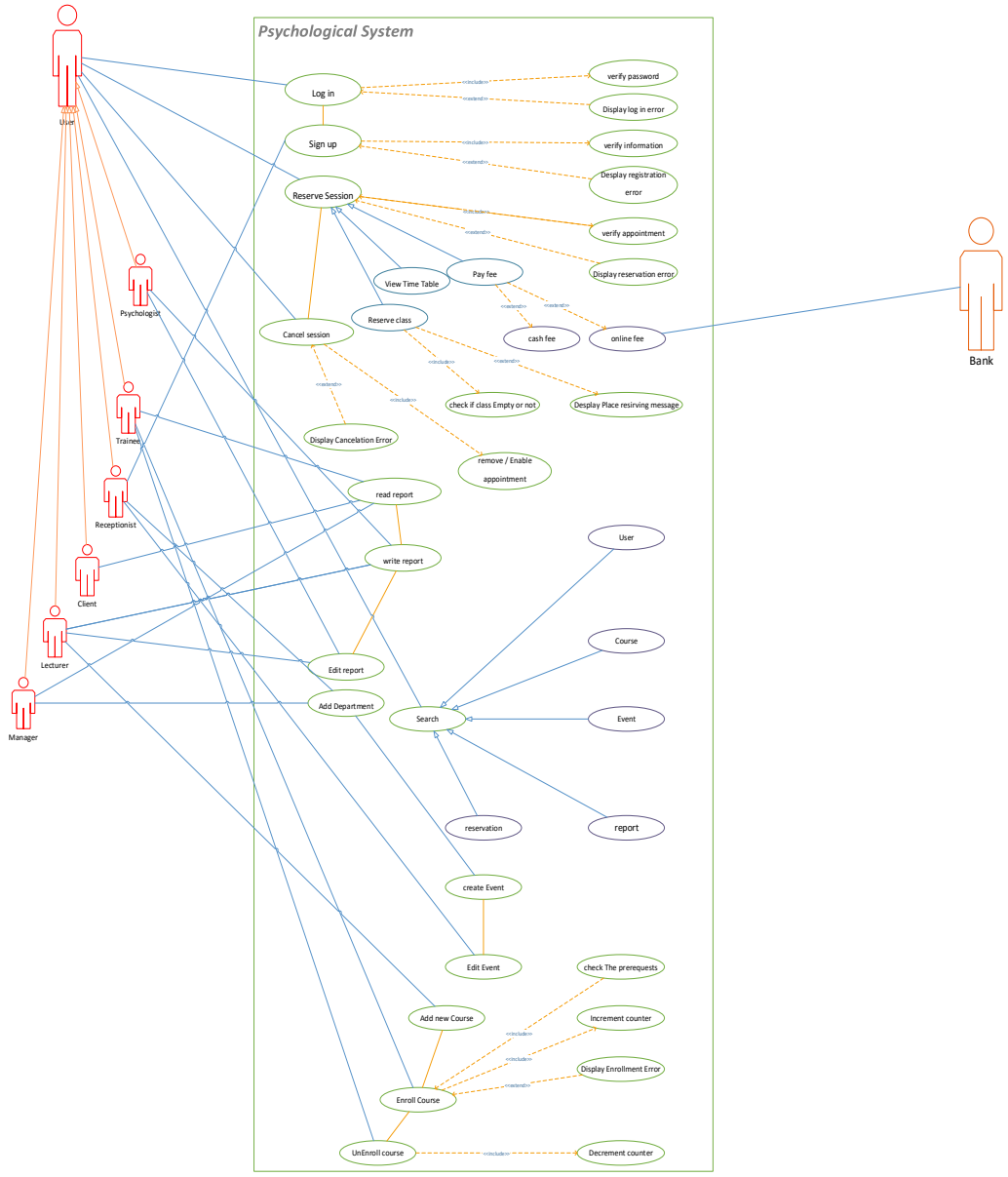
Context diagram



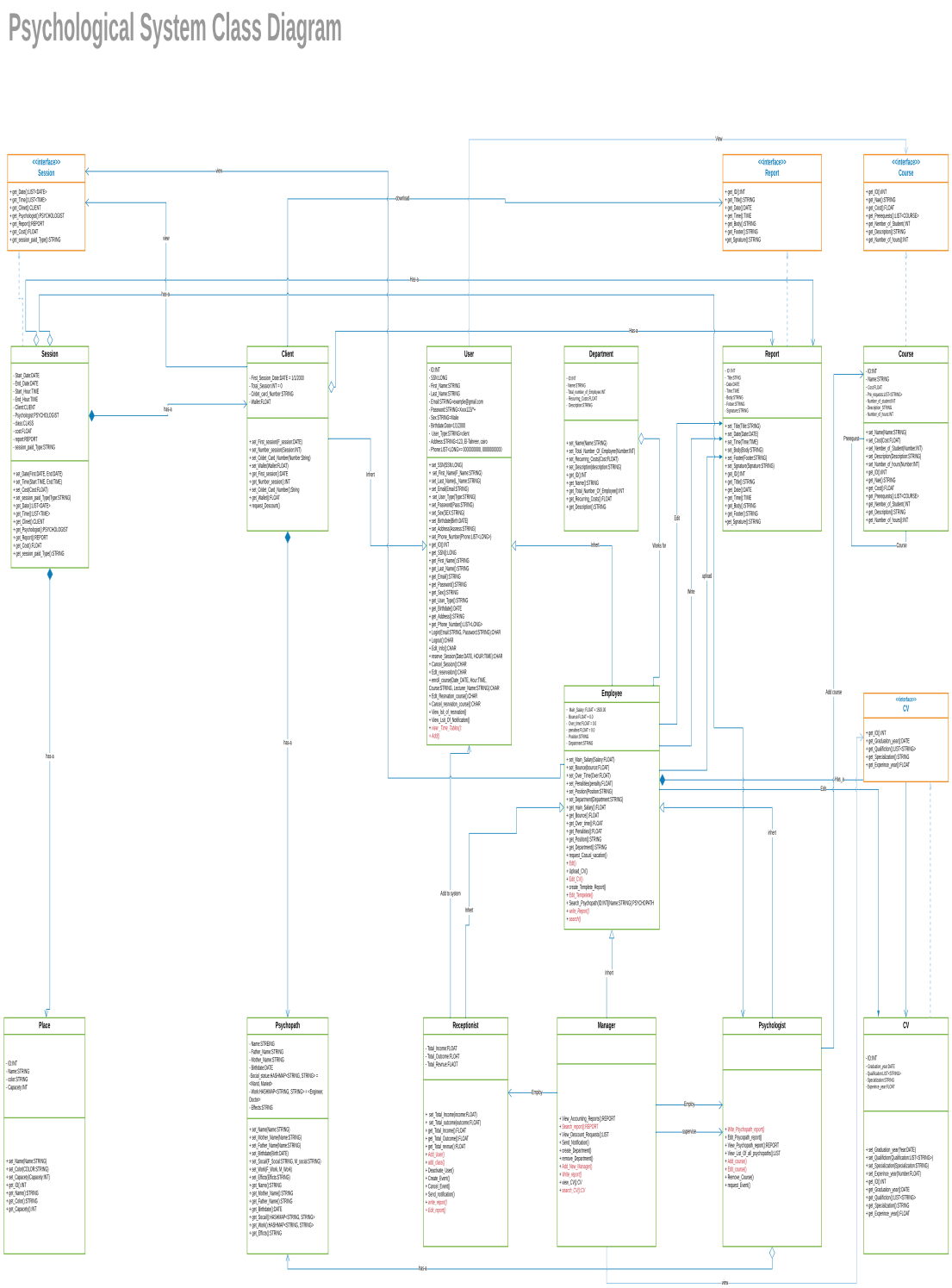
level one


Receptionist
Trainee
Psychologist
Lecturer
Manager
Client
Bank
1
Registration System
2
Accounting System
3
Reservation System
4
Notification System
5
Reports System
D1 Users' Data
D2 Accounts' Data
D2 Reservation Data
D2 reports Data
D2 reports Data
Log in
Log out
Register new user
Deactivate specific Account
Edit his Account and Save changes
Success/Fail Message
Saving Data
Verification Info.
Success/Fail Saving data
Success/Fail Verification
Search for specific User
Return searching results
Success/Fail Saving
Return total revenue of a day
Enter incoming and outcoming of a Day
Enter his Appointments
Edit Course information
Send his time table
Add Courses
Request pulling from credit card
Success/fail transaction
Success/Fail Enrollment and send his timetable
Enroll Course
Unenrolled Course
Accept/refuse request
Reserve Session
Request discounts to session
Cancel Session
Send Notifications
Send his timetable
Reserve Sessions
Enter his Appointments
Send reports of a day/week/month/year
Send reports of Week/ month/ year
Send Psychopath reports

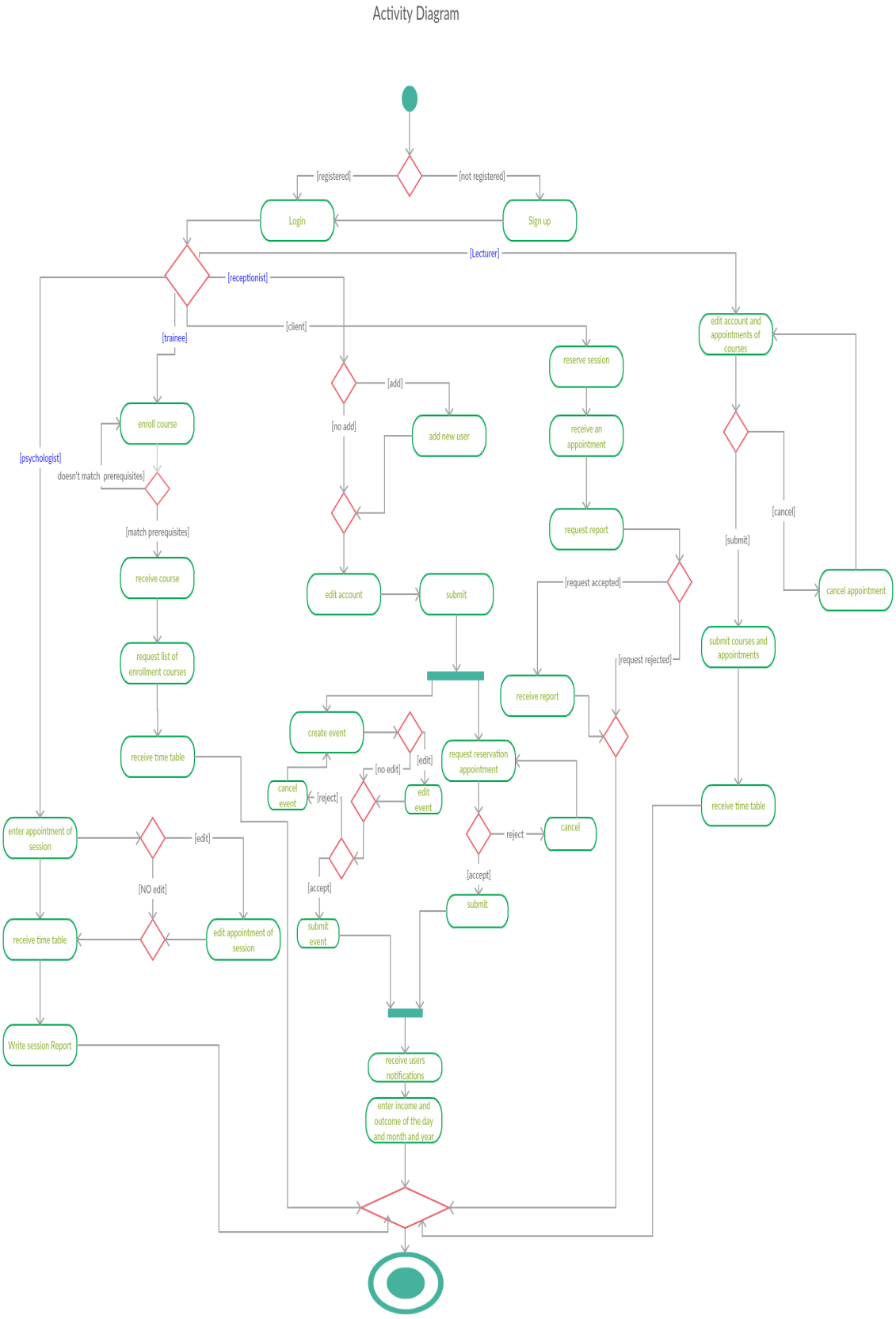
2.5.3 Use case diagram



2.5.4 Class diagram



2.5.5 Activity Diagram



2.6 ASSUMPTION DEPENDENCIES

Let us assume that this is an Inelegant psychological System and it is used in the following application:

2.6.1 For Management system

- Manager
  - connect with Employees and supervise them.
  - connect with clients and monitoring motions of organization.
  - veiw all reports that specified to Calculations and salaries of Employees and Exports resources ...etc.
- Receptionist stuff
  - create an account for new user.
  - make reservation of appointment or course to specific client.
  - recourd sessions per day and help you in the calculations.

2.6.2 For clinic system

- client:
  - A request for booking session with specific specialist.
  - choosing from available appointments for his specialist.
  - View his specific reports to his case and results, plans, and solving to issues that specified to his case.
  - update or cancel his resrvation, and update his profile.
- Specialist:
  - Enter his appointments to the system.
  - update and delete any appointment belongs to him, and update his profile.
  - upload the reports that specified to his cases and upload his plan of treatment.
  - connect with cases and receptionist.

2.6.3 For academic system

- Trainee:
  - A request for booking course.
  - choosing from available appointments for the course that he Enrolled it.
  - system should rtrive to him his schedule.

  - **update or cancel his resrvation, and update his profile.**

- **Trainer:**

  - **Enter his appointments for the courses to the system.**
  - **update and delete any appointment belongs to him, and update his profile.**
  - **upload the reports of result that specified to his trianees and upload scoring to them.**
  - **generate some tests and help trainer in correcting this tests.**

Chapter 3

SYSTEM FEATURES

3.1 reserve and booking

3.1.1 DESCRIPTION and PRIORITY

The Inelegant psychological System, classes of seats, personal preferences, prices, and bookings. Of course, this project has a high priority because it is very difficult to come to center without prior reservations.

3.1.2 Stimulus/Response Sequences

user should book reservation an appointment from choosing from available appointments and click to submit button to confirm the appointment

3.1.3 FUNCTIONAL REQUIREMENTS

- user should filling up simple application of information that are required.
- the Receptionists should create an account to the new user and send to him his user-name and his password.
- user after to receive his user name and his password should change and update his information.
- user can choose an appointment from available appointments and submit this appointment to save it in DB

3.2 Generate schedule and confirm appointments

3.2.1 DESCRIPTION and PRIORITY

after all psychologist Enter their appointments and users choose the appropriate his appointment and reserve it the system check in this appointments and if it has conflict appointments, system should solve it to use priority mechanism like oldest user, or first choose this appointment and so on .

3.2.2 Stimulus/Response Sequences

system generate schedule per day, take all appointments and date of the day and hours of all appointments and create the schedule.

3.2.3 FUNCTIONAL REQUIREMENTS

- system check all appointments and maintain them.
- system take all dates and hours of this appointments and put them in the schedule.
- system take the names of classes that will take sessions on it and put them in the schedule.
- system take specialists' name and client name and put them in the schedule.

3.3 Calculations system

3.3.1 DESCRIPTION and PRIORITY

calculations system help Receptionists stuff in some calculations like calculate the total incoming per day and week and month, calculate the total issued per day and week and month, and calculate total revenue per day and week and month, and calculate total worked hours per specialist to day and week and month.

3.3.2 Stimulus/Response Sequences

after the Receptionists enter data or details of sessions the system should calculate their calculations.

3.3.3 FUNCTIONAL REQUIREMENTS

- Receptionists should enter the fee of all sessions.
- system calculate the total incoming per day.
- system calculate the total issued per day.
- system calculate total revenue per day.
- system calculate total worked hours per psychologist to day
- system calculate the total incoming per week.
- system calculate the total issued per week.
- system calculate total revenue per week.
- system calculate total worked hours per psychologist to week.
- system calculate the total incoming per month.

- system calculate the total issued per month.
- system calculate total revenue per month.
- system calculate total worked hours per psychologist to month.

Chapter 4

Nonfunctional Requirements

4.1 Performance Requirements

our system has very fast performance We built it and design it to work very fast without any issues.

4.2 Safety Requirements

our system has no effects or dangerous on the users or machines because it is software system.

4.3 Security Requirements

we secure our system with the newest technology in security field to provide the best experience to the user without any fearing of security.

4.4 Software Quality Attributes

4.4.1 Availability

Availability defines the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period. Availability will be affected by system errors, infrastructure problems, malicious attacks, and system load. The key issues for availability are:

- A physical tier such as the database server or application server can fail or become unresponsive, causing the entire system to fail. Consider how to design fail-over support for the tiers in the system. For example, use Network Load Balancing for Web servers to distribute the load and prevent requests being directed to a server that is down. Also, consider using a RAID mechanism to mitigate system failure in the event of a disk failure. Consider if there is a need for a geographically separate redundant site to fail-over to in case of natural disasters such as earthquakes or tornado.

- Denial of Service (DoS) attacks, which prevent authorized users from accessing the system, can interrupt operations if the system cannot handle massive loads in a timely manner, often due to the processing time required, or network configuration and congestion. To minimize interruption from DoS attacks, reduce the attack surface area, identify malicious behavior, use application instrumentation to expose unintended behavior, and implement comprehensive data validation. Consider using the Circuit Breaker or Bulkhead patterns to increase system resiliency.

- Inappropriate use of resources can reduce availability. For example, resources acquired too early and held for too long cause resource starvation and an inability to handle additional concurrent user requests.

- Bugs or faults in the application can cause a system wide failure. Design for proper exception handling in order to reduce application failures from which it is difficult to recover.

- Frequent updates, such as security patches and user application upgrades, can reduce the availability of the system. Identify how you will design for run-time upgrades.

- A network fault can cause the application to be unavailable. Consider how you will handle unreliable network connections; for example, by designing clients with occasionally-connected capabilities.

- Consider the trust boundaries within your application and ensure that subsystems employ some form of access control or firewall, as well as extensive data validation, to increase resiliency and availability.

4.4.2 Conceptual Integrity

Conceptual integrity defines the consistency and coherence of the overall design. This includes the way that components or modules are designed, as well as factors such as coding style and variable naming. A coherent system is easier to maintain because you will know what is consistent with the overall design. Conversely, a system without conceptual integrity will constantly be affected by changing interfaces, frequently deprecating modules, and lack of consistency in how tasks are performed. The key issues for conceptual integrity are:

- Mixing different areas of concern within your design. Consider identifying areas of concern and grouping them into logical presentation, business, data, and service layers as appropriate.

- nconsistent or poorly managed development processes. Consider performing an Application Life-cycle Management (ALM) assessment, and make use of tried and tested development tools and methodologies.

- Lack of collaboration and communication between different groups involved in the application life-cycle. Consider establishing a development process integrated with tools to facilitate process work-flow, communication, and collaboration.

- Lack of design and coding standards. Consider establishing published guidelines for design and coding standards, and incorporating code reviews into your development process to ensure guidelines are followed.

- Existing (legacy) system demands can prevent both refactoring and progression toward a new platform or paradigm. Consider how you can create a migration path away from legacy technologies, and how to isolate applications from external dependencies. For example, implement the Gateway design pattern for integration with legacy systems.

4.4.3 Interoperability

Interoperability is the ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties. An interoperable system makes it easier to exchange and reuse information internally as well as externally. Communication protocols, interfaces, and data formats are the key considerations for interoperability. Standardization is also an important aspect to be considered when designing an interoperable system. The key issues for interoperability are:

- Interaction with external or legacy systems that use different data formats. Consider how you can enable systems to interoperate, while evolving separately or even being replaced. For example, use orchestration with adaptors to connect with external or legacy systems and translate data between systems; or use a canonical data model to handle interaction with a large number of different data formats.

- Boundary blurring, which allows artifacts from one system to defuse into another. Consider how you can isolate systems by using service interfaces and/or mapping layers. For example, expose services using interfaces based on XML or standard types in order to support interoperability with other systems. Design components to be cohesive and have low coupling in order to maximize flexibility and facilitate replacement and reusability.

- Lack of adherence to standards. Be aware of the formal and de facto standards for the domain you are working within, and consider using one of them rather than creating something new and proprietary.

4.4.4 Maintainability

Maintainability is the ability of the system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the application's functionality in order to fix errors, or to meet new business requirements. Maintainability can also affect the time it takes to restore the system to its operational status following a failure or removal from operation for an upgrade. Improving system maintainability can increase availability and reduce the effects of run-time defects. An application's maintainability is often a

function of its overall quality attributes but there a number of key issues that can directly affect maintainability:

- Excessive dependencies between components and layers, and inappropriate coupling to concrete classes, prevents easy replacement, updates, and changes; and can cause changes to concrete classes to ripple through the entire system. Consider designing systems as well-defined layers, or areas of concern, that clearly delineate the system's UI, business processes, and data access functionality. Consider implementing cross-layer dependencies by using abstractions (such as abstract classes or interfaces) rather than concrete classes, and minimize dependencies between components and layers.

- The use of direct communication prevents changes to the physical deployment of components and layers. Choose an appropriate communication model, format, and protocol. Consider designing a pluggable architecture that allows easy upgrades and maintenance, and improves testing opportunities, by designing interfaces that allow the use of plug-in modules or adapters to maximize flexibility and extensibility.

- Reliance on custom implementations of features such as authentication and authorization prevents reuse and hampers maintenance. To avoid this, use the built-in platform functions and features wherever possible.

- The logic code of components and segments is not cohesive, which makes them difficult to maintain and replace, and causes unnecessary dependencies on other components. Design components to be cohesive and have low coupling in order to maximize flexibility and facilitate replacement and reusability.

- The code base is large, unmanageable, fragile, or over complex; and refactoring is burdensome due to regression requirements. Consider designing systems as well defined layers, or areas of concern, that clearly delineate the system's UI, business processes, and data access functionality. Consider how you will manage changes to business processes and dynamic business rules, perhaps by using a business workflow engine if the business process tends to change. Consider using business components to implement the rules if only the business rule values tend to change; or an external source such as a business rules engine if the business decision rules do tend to change.

- The existing code does not have an automated regression test suite. Invest in test automation as you build the system. This will pay off as a validation of the system's functionality, and as documentation on what the various parts of the system do and how they work together.

- Lack of documentation may hinder usage, management, and future upgrades. Ensure that you provide documentation that, at minimum, explains the overall structure of the application.

4.4.5 Manageability

Manageability **defines how easy it is for system administrators to manage the application, usually through sufficient and useful instrumentation exposed for use in monitoring systems and for debugging and performance**

tuning. Design your application to be easy to manage, by exposing sufficient and useful instrumentation for use in monitoring systems and for debugging and performance tuning. The key issues for manageability are:

- Lack of health monitoring, tracing, and diagnostic information. Consider creating a health model that defines the significant state changes that can affect application performance, and use this model to specify management instrumentation requirements. Implement instrumentation, such as events and performance counters, that detects state changes, and expose these changes through standard systems such as Event Logs, Trace files, or Windows Management Instrumentation (WMI). Capture and report sufficient information about errors and state changes in order to enable accurate monitoring, debugging, and management. Also, consider creating management packs that administrators can use in their monitoring environments to manage the application.

- Lack of runtime configurability. Consider how you can enable the system behavior to change based on operational environment requirements, such as infrastructure or deployment changes.

- Lack of troubleshooting tools. Consider including code to create a snapshot of the system's state to use for troubleshooting, and including custom instrumentation that can be enabled to provide detailed operational and functional reports. Consider logging and auditing information that may be useful for maintenance and debugging, such as request details or module outputs and calls to other systems and services.

-

4.4.6 Performance

Performance is an indication of the responsiveness of a system to execute specific actions in a given time interval. It can be measured in terms of latency or throughput. Latency is the time taken to respond to any event. Throughput is the number of events that take place in a given amount of time. An application's performance can directly affect its scalability, and lack of scalability can affect performance. Improving an application's performance often improves its scalability by reducing the likelihood of contention for shared resources. Factors affecting system performance include the demand for a specific action and the system's response to the demand. The key issues for performance are:

- Increased client response time, reduced throughput, and server resource over utilization. Ensure that you structure the application in an appropriate way and deploy it onto a system or systems that provide sufficient resources. When communication must cross process or tier boundaries, consider using coarse-grained interfaces that require the minimum number of calls (preferably just one) to execute a specific task, and consider using asynchronous communication.

- Increased memory consumption, resulting in reduced performance, excessive cache misses (the inability to find the required data in the cache), and increased data store access. Ensure that you design an efficient and appropriate caching strategy.

- Increased database server processing, resulting in reduced throughput. Ensure that you choose effective types of transactions, locks, threading, and queuing approaches. Use efficient queries to minimize performance impact, and avoid fetching all of the data when only a portion is displayed. Failure to design for efficient database processing may incur unnecessary load on the database server, failure to meet performance objectives, and costs in excess of budget allocations.

- Increased network bandwidth consumption, resulting in delayed response times and increased load for client and server systems. Design high performance communication between tiers using the appropriate remote communication mechanism. Try to reduce the number of transitions across boundaries, and minimize the amount of data sent over the network. Batch work to reduce calls over the network.

4.4.7 Reliability

Reliability is the ability of a system to continue operating in the expected way over time. Reliability is measured as the probability that a system will not fail and that it will perform its intended function for a specified time interval. The key issues for reliability are:

- The system crashes or becomes unresponsive. Identify ways to detect failures and automatically initiate a failover, or redirect load to a spare or backup system. Also, consider implementing code that uses alternative systems when it detects a specific number of failed requests to an existing system.

- Output is inconsistent. Implement instrumentation, such as events and performance counters, that detects poor performance or failures of requests sent to external systems, and expose information through standard systems such as Event Logs, Trace files, or WMI. Log performance and auditing information about calls made to other systems and services.

- The system fails due to unavailability of other externalities such as systems, networks, and databases. Identify ways to handle unreliable external systems, failed communications, and failed transactions. Consider how you can take the system offline but still queue pending requests. Implement store and forward or cached message-based communication systems that allow requests to be stored when the target system is unavailable, and replayed when it is online. Consider using Windows Message Queuing or BizTalk Server to provide a reliable once-only delivery mechanism for asynchronous requests.

4.4.8 Reusability

Reusability is the probability that a component will be used in other components or scenarios to add new functionality with little or no change.

Reusability minimizes the duplication of components and the implementation time. Identifying the common attributes between various components is the first step in building small reusable components for use in a larger system. The key issues for reusability are:

- The use of different code or components to achieve the same result in different places; for example, duplication of similar logic in multiple components, and duplication of similar logic in multiple layers or subsystems. Examine the application design to identify common functionality, and implement this functionality in separate components that you can reuse. Examine the application design to identify crosscutting concerns such as validation, logging, and authentication, and implement these functions as separate components.
- The use of multiple similar methods to implement tasks that have only slight variation. Instead, use parameters to vary the behavior of a single method.
- Using several systems to implement the same feature or function instead of sharing or reusing functionality in another system, across multiple systems, or across different subsystems within an application. Consider exposing functionality from components, layers, and subsystems through service interfaces that other layers and systems can use. Use platform agnostic data types and structures that can be accessed and understood on different platforms.

4.4.9 Scalability

Scalability **is ability of a system to either handle increases in load without impact on the performance of the system, or the ability to be readily enlarged. There are two methods for improving scalability: scaling vertically (scale up), and scaling horizontally (scale out). To scale vertically, you add more resources such as CPU, memory, and disk to a single system. To scale horizontally, you add more machines to a farm that runs the application and shares the load. The key issues for scalability are:**

- Applications cannot handle increasing load. Consider how you can design layers and tiers for scalability, and how this affects the capability to scale up or scale out the application and the database when required. You may decide to locate logical layers on the same physical tier to reduce the number of servers required while maximizing load sharing and failover capabilities. Consider partitioning data across more than one database server to maximize scale-up opportunities and allow flexible location of data subsets. Avoid stateful components and subsystems where possible to reduce server affinity.
- Users incur delays in response and longer completion times. Consider how you will handle spikes in traffic and load. Consider implementing code that uses additional or alternative systems when it detects a predefined service load or a number of pending requests to an existing system.
- The system cannot queue excess work and process it during periods of reduced load. Implement store-and-forward or cached message-based communication systems that allow requests to be stored when the target system is unavailable, and replayed when it is online.

4.4.10 Security

Security is the capability of a system to reduce the chance of malicious or accidental actions outside of the designed usage affecting the system, and prevent disclosure or loss of information. Improving security can also increase the reliability of the system by reducing the chances of an attack succeeding and impairing system operation. Securing a system should protect assets and prevent unauthorized access to or modification of information. The factors affecting system security are confidentiality, integrity, and availability. The features used to secure systems are authentication, encryption, auditing, and logging. The key issues for security are:

- Spoofing of user identity. Use authentication and authorization to prevent spoofing of user identity. Identify trust boundaries, and authenticate and authorize users crossing a trust boundary.
- Damage caused by malicious input such as MYSQL injection and cross-site scripting. Protect against such damage by ensuring that you validate all input for length, range, format, and type using the constrain, reject, and sanitize principles. Encode all output you display to users.
- Data tampering. Partition the site into anonymous, identified, and authenticated users and use application instrumentation to log and expose behavior that can be monitored. Also use secured transport channels, and encrypt and sign sensitive data sent across the network.
- Repudiation of user actions. Use instrumentation to audit and log all user interaction for application critical operations.
- Information disclosure and loss of sensitive data. Design all aspects of the application to prevent access to or exposure of sensitive system and application information.
- Interruption of service due to DoS attacks. Consider reducing session timeouts and implementing code or hardware to detect and mitigate such attacks.

4.4.11 Supportability

Supportability is the ability of the system to provide information helpful for identifying and resolving issues when it fails to work correctly. The key issues for supportability are:

- ack of diagnostic information. Identify how you will monitor system activity and performance. Consider a system monitoring application, such as Microsoft System Center.
- Lack of troubleshooting tools. Consider including code to create a snapshot of the system's state to use for troubleshooting, and including custom instrumentation that can be enabled to provide detailed operational and functional reports. Consider logging and auditing information that may be useful for maintenance and debugging, such as request details or module outputs and calls to other systems and services.

- Lack of tracing ability. Use common components to provide tracing support in code, perhaps though Aspect Oriented Programming (AOP) techniques or dependency injection. Enable tracing in Web applications in order to troubleshoot errors.

- Lack of health monitoring. Consider creating a health model that defines the significant state changes that can affect application performance, and use this model to specify management instrumentation requirements. Implement instrumentation, such as events and performance counters, that detects state changes, and expose these changes through standard systems such as Event Logs, Trace files, or WMI.

- Capture and report sufficient information about errors and state changes in order to enable accurate monitoring, debugging, and management. Also, consider creating management packs that administrators can use in their monitoring environments to manage the application.

4.4.12 Testability

Testability is a measure of how well system or components allow you to create test criteria and execute tests to determine if the criteria are met. Testability allows faults in a system to be isolated in a timely and effective manner. The key issues for testability are:

- Complex applications with many processing permutations are not tested consistently, perhaps because automated or granular testing cannot be performed if the application has a monolithic design. Design systems to be modular to support testing. Provide instrumentation or implement probes for testing, mechanisms to debug output, and ways to specify inputs easily. Design components that have high cohesion and low coupling to allow testability of components in isolation from the rest of the system.

- Lack of test planning. Start testing early during the development life cycle. Use mock objects during testing, and construct simple, structured test solutions.

- Poor test coverage, for both manual and automated tests. Consider how you can automate user interaction tests, and how you can maximize test and code coverage.

- Input and output inconsistencies; for the same input, the output is not the same and the output does not fully cover the output domain even when all known variations of input are provided. Consider how to make it easy to specify and understand system inputs and outputs to facilitate the construction of test cases.

4.4.13 User Experience / Usability

User Experience / Usability The application interfaces must be designed with the user and consumer in mind so that they are intuitive to use, can

be localized and globalized, provide access for disabled users, and provide a good overall user experience. The key issues for user experience and usability are:

- too much interaction (an excessive number of clicks) required for a task. Ensure you design the screen and input flows and user interaction patterns to maximize ease of use.
- Incorrect flow of steps in multistep interfaces. Consider incorporating workflows where appropriate to simplify multistep operations.
- Data elements and controls are poorly grouped. Choose appropriate control types (such as option groups and check boxes) and lay out controls and content using the accepted UI design patterns.
- Feedback to the user is poor, especially for errors and exceptions, and the application is unresponsive. Consider implementing technologies and techniques that provide maximum user interactivity, such as Asynchronous JavaScript and XML (AJAX) in Web pages and client-side input validation. Use asynchronous techniques for background tasks, and tasks such as populating controls or performing long-running tasks.

Acronyms

AJAX Asynchronous JavaScript And XML 30
ALM Application Life-cycle Management 22
AOP Aspect Oriented Programming 29
DB Database 5, 17
DBMS Database Management System 6
DOS Denial Of Service 22, 28
JSP Java Server Pages 8
SRS Software Requirements Specification 5
SSN Social Security Number 5
WMI Windows Management Instrumentation 25, 29

Glossary

Availability	the proportion of time that the system is functional and working. 21
behavior	Everything we do that can be directory observed 1
Conceptual integrity	the consistency and coherence of the overall design. 22
database	is a collection of related data 7, 8
entity relationship model	The ER model describes data as entities,relationships, and attributes 7
Interoperability	is the ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties. 23
Maintainability	is the ability of the system to under go changes with a degree of ease. 23
Manageability	how easy it is for system administrators to manage the application, usually through sufficient and useful instrumentation exposed for use in monitoring systems and for debugging and performance tuning. 24
mental processes	The thoughts, felling and motives that each of us experiences privately but that can not be observed directly 2
MYSQL	the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. 28
Performance	s an indication of the responsiveness of a system to execute specific actions in a given time interval. 25
phenomena	A fact or situation that is observed to exist or happen, especially one whose cause or explanation is in question. 1
psychologist	Who the person that do analyze people's problems and try to help them cope more effectively 1, 3, 17, 18, 19
psychology	Is the scientific study of behavior and mental processes 1, 2
Receptionists	are usually seated at the entrance of an office and do a variety of administrative tasks including, but not limited to, answering phones calls, making photocopies, distributing mail, signing for packages, and general office upkeep. Some companies employ a receptionist for the sole purpose of answering phones, but with the advancement of technology, digital answering services and outsourced receptionist firms are the new wave of the future. 17, 18

Reliability	s the ability of a system to continue operating in the expected way over time. 26
Reusability	is the probability that a component will be used in other components or scenarios to add new functionality with little or no change. 26
Scalability	is ability of a system to either handle increases in load without impact on the performance of the system, or the ability to be readily enlarged. 27
Security	is the capability of a system to reduce the chance of malicious or accidental actions outside of the designed usage affecting the system, and prevent disclosure or loss of information. 28
Supportability	is the ability of the system to provide information helpful for identifying and resolving issues when it fails to work correctly. 28
Testability	is a measure of how well system or components allow you to create test criteria and execute tests to determine if the criteria are met. 29
User Experience / Usability	The application interfaces must be designed with the user and consumer in mind so that they are intuitive to use, can be localized and globalized, provide access for disabled users, and provide a good overall user experience. 29

Bibliography

[1] Ramez Elmasri and Shamkant B Navathe. *Fundamentals of database systems*. Pearson, 2015.

[2] John W Santrock. *Psychology: Essentials*. McGraw-Hill Boston, 2003.