Definition of SDLC

The Software Development Life Cycle (SDLC), a structured process, offers the quickest possible delivery of high-quality, affordable software. The SDLC's goal is to create exceptional software that meets and exceeds all client expectations and objectives. (synopsys, n.d.)

- Planning: The planning stage of the SDLC is where the project is identified and the objectives of the project are outlined. During this stage, the scope of the project is identified and the team members are chosen. The project timeline and budget are also established.

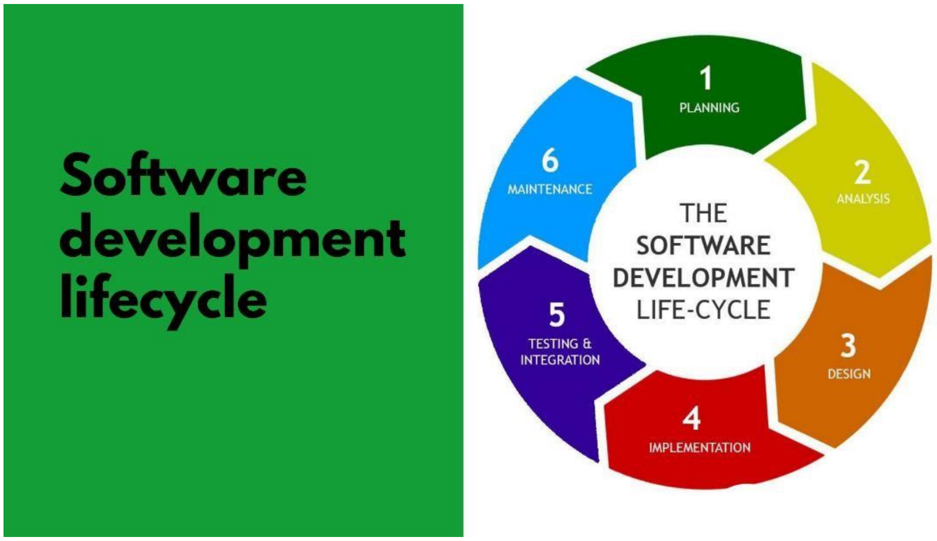
- Analysis: During the analysis stage of the SDLC, the requirements for the project are identified and documented. This is done by gathering data from the stakeholders and analyzing the data. The team also creates a design document that outlines how the system will work.

- Design: The design stage of the SDLC is where the team develops the architecture for the system and creates a detailed design document. This document outlines the components of the system and how they interact with each other.

- Development: During the development stage of the SDLC, the team implements the design and develops the system. This is done by writing code, creating databases, and setting up the infrastructure.

- Testing: During the testing stage of the SDLC, the system is tested to make sure it meets the requirements outlined in the design document. This includes unit testing, system testing, and user acceptance testing.

- Maintenance: The maintenance stage of the SDLC is where the system is monitored and maintained to make sure it is up to date and running properly. This includes making sure the system is patched, updated, and monitored for performance and security.



Web based appointment system for the hospital with SDLC model

1. Planning - Making a plan is important in the making of software for hospitals. The plan involves figuring out what the hospital needs, who is involved in the project, and what resources are available. The first step is to understand the hospital's needs, like what types of appointments they have and what other services they want. Next, the people involved in the project, like the IT team and hospital staff, are identified. Then, the resources, like the budget and technology, are looked at. After that, a plan is made with a timeline, tasks, and other important information, and shared with everyone involved. Finally, the software is designed, with user-interfaces, connections to the hospital's systems, and testing.

2. Analysis - The second step in making a new software system for a hospital is to figure out what it needs to do. The system should know about different types of appointments (like routine, emergency, or follow-up visits) and who might use it (patients, staff, etc.). It should also have features to let patients make appointments, let staff manage appointments, and let patients keep their information safe. Finally, the system should think about who will use it (staff, patients, etc.) and what devices they might use.

3. Design - The Design step in creating a website for scheduling appointments at a hospital involves making a plan. The plan includes how the website will look, how it will keep

information safe, how it will work with other systems, and what it will be able to do. The team also thinks about if the website can handle more users and if it will need changes in the future. After the plan is ready, the people making the website can start building it.

4. Development - The process of making a web-based appointment system for a hospital goes through several steps. First, they plan and design what the system should do and how it should look. Then they write the computer code to make the system work. After that, they test it to make sure everything is working correctly. They fix any problems they find during testing. Finally, they make the system available for people to use

5. Testing - Testing is important for making sure a hospital's online appointment system works well. This means checking if the system is accurate, easy to use, secure, and can handle a lot of appointments and patient information. To do this, the system will go through 3 steps of testing. First, they will test how easy it is to use the website. Second, they will check how well the system can store and retrieve information. Finally, they will make sure the system can handle a lot of appointments and work well even if something goes wrong.

6. Maintenance - To keep a web-based hospital appointment system running smoothly and meeting the needs of users, it's important to regularly check for problems and fix them. This means monitoring the system's performance, making sure it works well with other systems, and keeping its security up-to-date. It's also important to check that the system still works well for users and meets any legal requirements. The team in charge of maintenance should be ready to fix any problems that come up and make changes as needed. Keeping up with maintenance is an ongoing task to ensure the system runs well and users are happy.

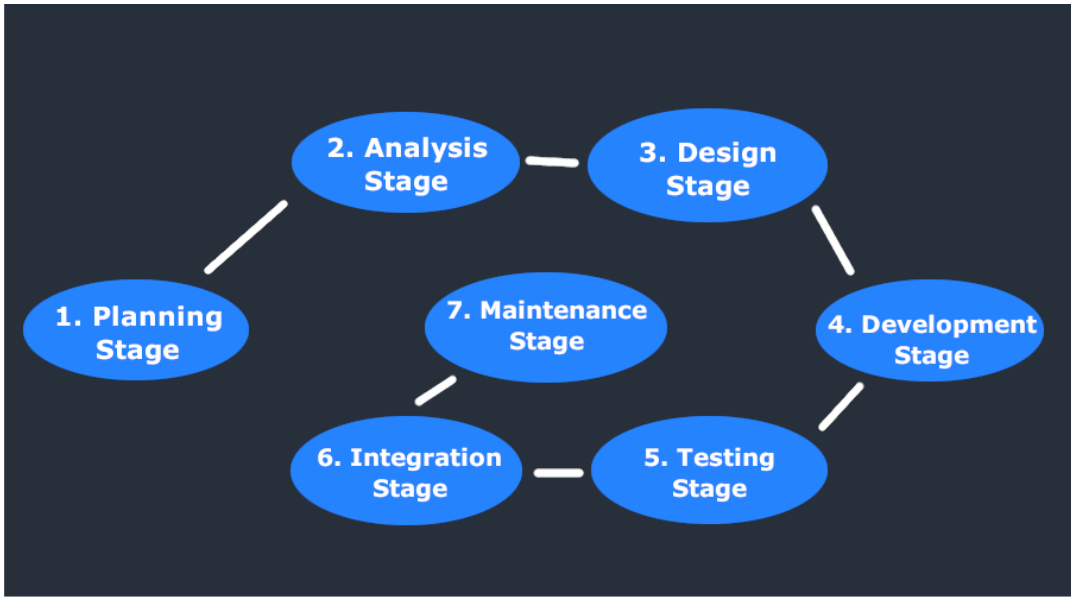
Models of SDLC

1. Waterfall Model:

The Conditions The Waterfall Model is a sequential software development method in which development is represented as flowing downward like a waterfall. It includes the phases of Gathering, Design, Implementation, Testing, Integration, and Maintenance.

The five processes that make up this methodology are requirements analysis and specification, design, implementation, unit testing, integration and system testing, and operation and maintenance.

The steps never overlap and are always performed in this order. Prior to continuing with the next The previous step must be finished by the developer. The diagrammatic representation of this model is known as the "Waterfall Model" because of how much it resembles a waterfall cascade. Below is a list of the steps in the waterfall model.



This paradigm is straightforward to use and uses minimal resources. The objectives are clear and consistent, and they hold true as the project moves forward. Each phase's starting and ending positions are specified, making it simple to keep track of development. Before development begins, the release date and final price of the finished product can be set. Customers can easily handle and understand their data thanks to robust reporting tools.

2. Spiral Model:

The Spiral Model is a software development process that includes elements of both design and prototyping-in-stages, combining the advantages of top-down and bottom-up ideas. The spiral strategy is intended for large, expensive, and difficult businesses.

This risk-driven approach incorporates elements of iterative and waterfall modeling. It makes it possible for new software versions to be developed quickly. Through a series of incremental releases, the program is developed utilizing the spiral model. The concept is initially developed as a paper model or a prototype in the early cycles or iterations. The program is implemented in successive cycles of the cycle as extended or operational prototypes. For initiatives that need to develop over time, the spiral model is utilized. The phases of each cycle are as follows.

- Identifying options and arriving at a fair judgment.
- Identify and manage risks.
- Create the project's upcoming iteration.
- Look over and get ready for the next step.

3. V-Model:

The V-Model is a graphical representation of a software development process. It makes a link between each step of the development life cycle and its earlier and later stages and is based on the waterfall model.

The V-model is an SDLC framework in which steps are completed consecutively and in the shape of a V. The Verification and Validation Model is another name for it.
Each important development stage is linked to a testing phase in the V-Paradigm, a development technique that expands on the waterfall model. This implies that there is a testing phase that is closely related to each stage of the development cycle. In this highly organized strategy, the next phase doesn't begin until the preceding phase has ended. Under this methodology, phases are completed one at a time, which calls for severe discipline.

4. Agile Model:

A gradual and iterative software development process is the Agile Model. It is a flexible, lightweight technique that is concentrated on producing high-caliber software quickly. Collaboration amongst self-organizing, cross-functional teams serves as its foundation

The agile development paradigm is also a part of the incremental model.
In software development, small, incremental steps are taken.
As a result, the application is updated gradually based on previous modifications.
To maintain the program's caliber, every version is put through a thorough testing process.

It is used in time-sensitive applications.

An agile model has the following steps.
Gathering requirements, requirement analysis, design, coding, unit testing, and acceptance testing

The foundational piece of agile is the sprint. The team concentrates on delivering a certain set of product features during a sprint. This minimizes documentation while including the client in all phases of the development process. Only when it is absolutely necessary is documentation created; otherwise, informal communication is used. This approach is excellent for projects when the demands change, but it should not be used for large-scale initiatives. Agile development approaches include SCRUM, Extreme Programming (XP), Crystal, Atern, and Kanban, to name a few.

5. .RAD model

The RAD model is an adaptation of agile methodology that places an emphasis on continuous prototyping and iterations that are affected by user feedback. Instead of adhering to a rigid development strategy, it enables you to add updates based on usage. They are able to manage new requirements as they arise without having an impact on the process as a whole based on client or user feedback. It now results in speedy prototypes for testing and additional improvements. The RAD model controls the development process utilizing 5 fundamental steps or processes, despite allowing for infinite flexibility and adjustments.

6. Iterative model

Using this method, a minimal set of criteria can be used to construct the first version of the software. A new version of the software is created utilizing a new iteration if adjustments to the initial version are required. Each release is completed by the iterative model over a predetermined length of time called an iteration. Once the stakeholders are happy with the outcome and no additional iterations are required, these iterations are repeated repeatedly. The steps for one iteration are as follows.

7. Incremental model

The model's requirements are divided into a number of increments, each of which is provided to the client for review and comment. Before the final product is developed, it allows the stakeholders to make any changes that suit their tastes. With each iteration, this model passes through the steps of requirements, design, development, and testing.

Suggested SDLC model

Hemas' Hospital is a hospital in Wattala that offers both medical and surgical services. For hospital management, there is now a subpar online information system. The system can only be used to schedule doctor appointments. They require a better and more efficient method because the hospital is seeing more patients.
The hospital has a comprehensive list of specifications for the new system. Because the Agile development methodology was so advantageous for our use case, we chose to implement it. The most successful agile approach is the SCRUM Agile methodology. The rapid and regular inspections will enable the final product to be precisely customized for the needs of the client.

Mapping the suggested solution to the chosen SDLC model

Needs for gathering: In order to establish a project scope and try to understand their demands, we visited Hemas Hospital during this step. With the stakeholders of the healthcare center, we also spoke about the budget and project plan. We were able to learn the following facts:

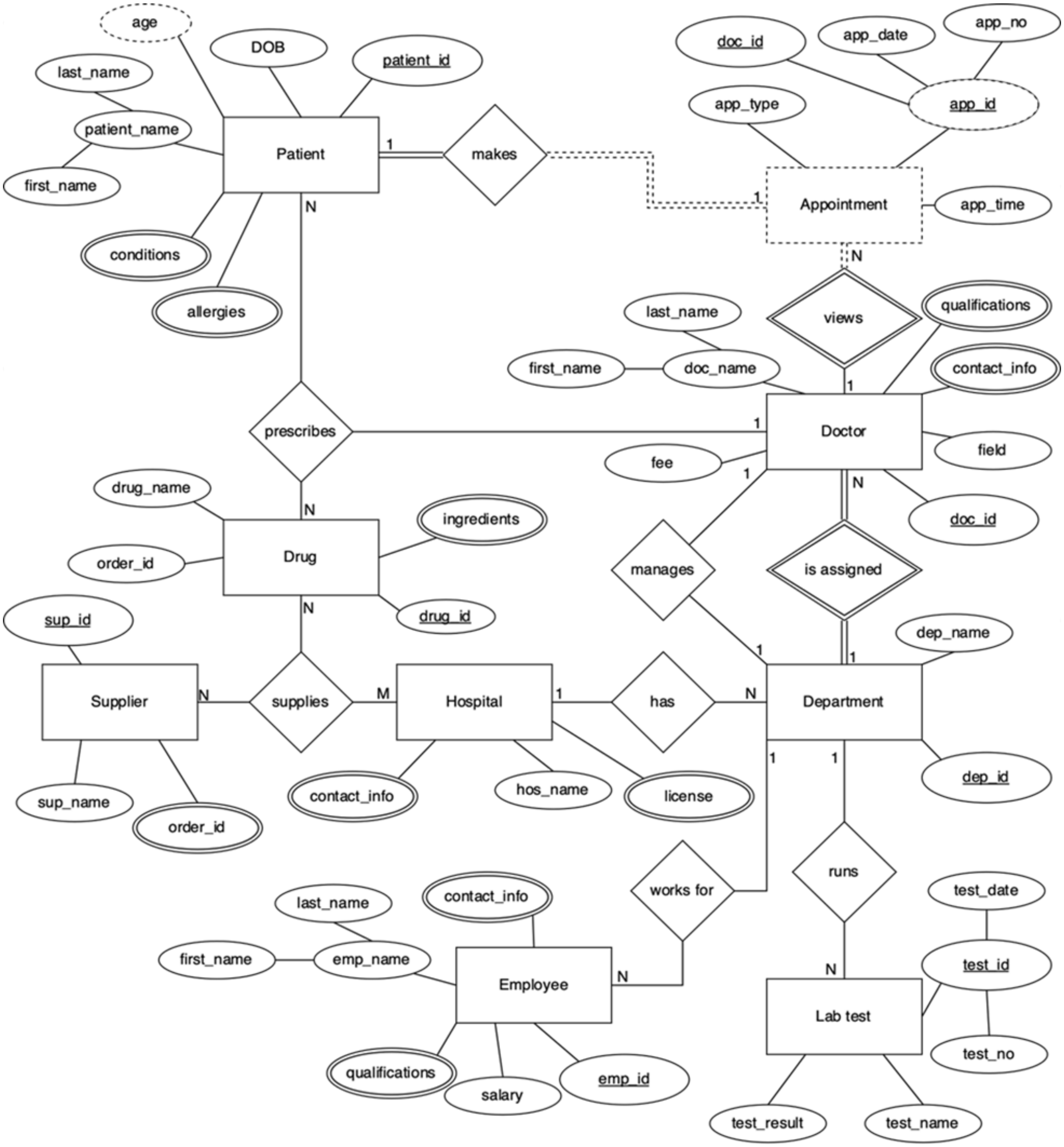
- Specifics regarding the hospital

Patient data, staff and medical professional data, and system analysis
Examination of Requirements Throughout the process of gathering requirements, we were able to learn a lot about the client's requirements. The hospital management information system was the biggest problem because it was antiquated and poorly constructed. So we proposed creating a better system with additional features. As the system is being developed, we talked about the changes that need to be made. Following the completion of the requirements, the software requirement

Advantages of Using the Agile SDLC Model for a Hospital's Web-Based Appointment System.

Adaptable and responsive to changes, the Agile SDLC model is a viable option for web-based appointment systems used in hospitals. Customers can provide input often under the Agile model, which helps the system adapt to the demands of the institution. The Agile paradigm is appropriate for web-based projects as well since it enables speedy system development and simple system updates. It promotes developer cooperation, creating a productive team. By locating and fixing issues early in the development cycle, the Agile model can reduce development costs while also guaranteeing that the system will satisfy the hospital's needs within the parameters of time and money. Overall, the Agile SDLC approach can produce a system that is successful, effective, and user-friendly.

Relational Data Model



Data Dictionaries

Patients

Serial number	attribute	name	Data type	No.of characters
01	Patient_ id	Patient identifier	String	10
02	Patient_ name	Patient name	String	100
03	age	Age of the Patient	Integer	02
04	conditions	Medical Condition	String	100
05	allergies	Allergic Conditions	String	100

Doctor

Serial number	attribute	name	Data type	No.of characters
01	doc_name	Name of Doctor	String	50
02	doc_id	Doctor identifier	String	10
03	fee	Charges for consultation	Integer	06
04	field	The doctor's specialization	String	50
05	qualifications	The doctor's qualifications	String	50
06	contact _info	Contact details of doctor	String	10

Department

Serial number	attribute	name	Data type	No.of characters
01	dep_name	Name of department	String	20
02	dep_id	Department identifier	String	10

Appointment

Serial number	**attribute**	**name**	**Data type**	**No.of characters**
01	app_type	Type of appointment	Char	10
02	app_id	Appointment identifier	String	10
03	app_time	Time of the appointment	Int	04
04	app_date	Date of the appointment	String	10
05	app_no.	Number of the appointment	Int	02
06	doc_id	Department identifier	String	10

Hospital

Serial number	**attribute**	**name**	**Data type**	**No.of characters**
01	hos_name	Name of hospital	string	20
02	Contact_info.	Hospital contact info	string	10
03	license	Medical licenses of the hospital	string	10

Employees

Serial number	**attribute**	**name**	**Data type**	**No.of characters**
01	emp_name	Name of employee	String	50

02	emp_id	Employee identifier	String	10
03	salary	The employee's salary	Int	06
04	qualifications	The employee's qualifications	String	50
05	contact_info	Contact details of employee	String	10

Lab test

Serial number	**attribute**	**name**	**Data type**	**No.of characters**
01	test_name	Name of test	string	20
02	test_id	Test identifier	string	10
03	test_results	Result of the test	boolean	04
04	test_date	Date of the test	string	10
05	test _no.	Number of the test	int	03

Suppliers

Serial number	**attribute**	**name**	**Data type**	**No.of characters**
01	sup_name	Name of supplier	string	20
02	sup_id	Supplier identifier	string	10
03	order_id	Order identifier	string	10

Drug

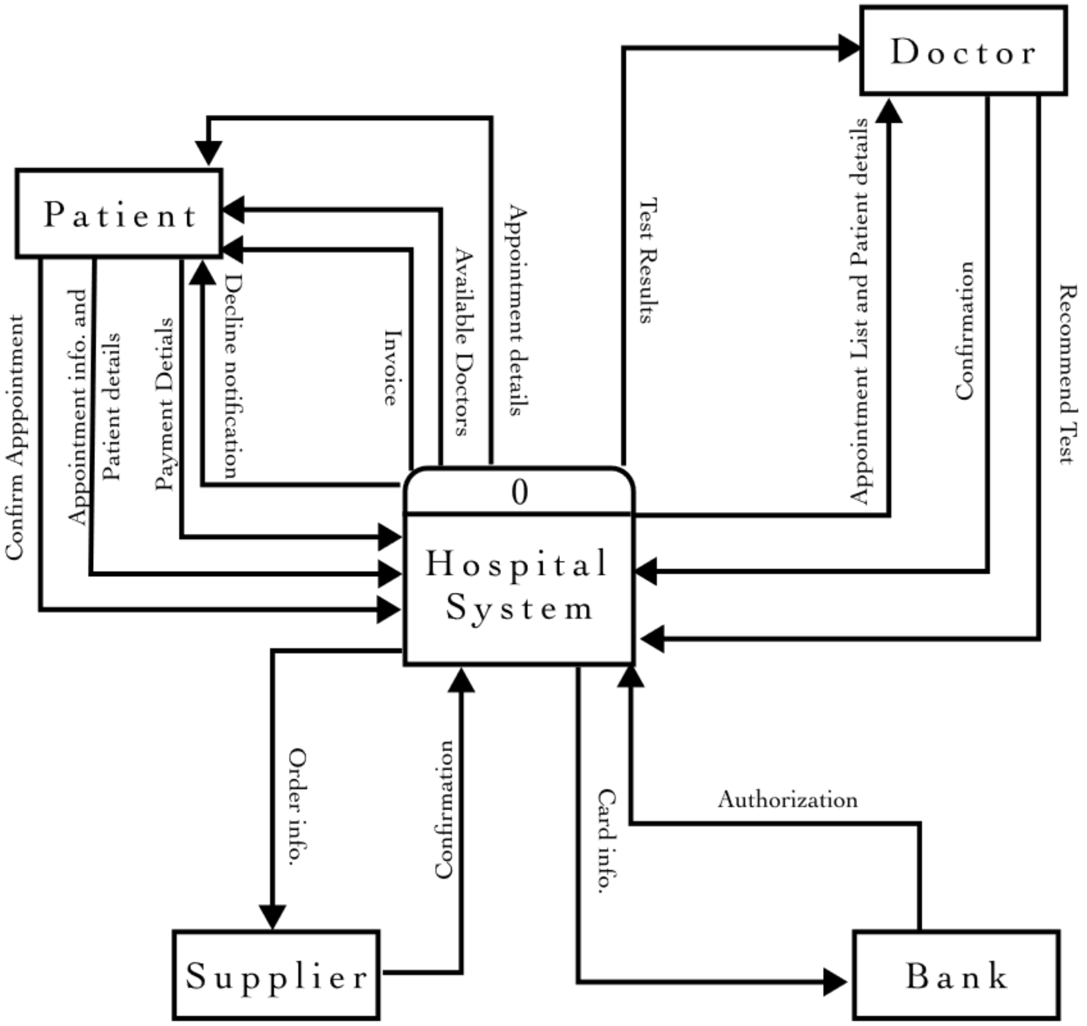
Serial number	**attribute**	**name**	**Data type**	**No.of characters**
01	drug_name	Name of drug	String	20
02	drug_id	Drug identifier	String	10
03	order_id	Order identifier	String	10

04	ingredients	Ingredients in the drug	String	50

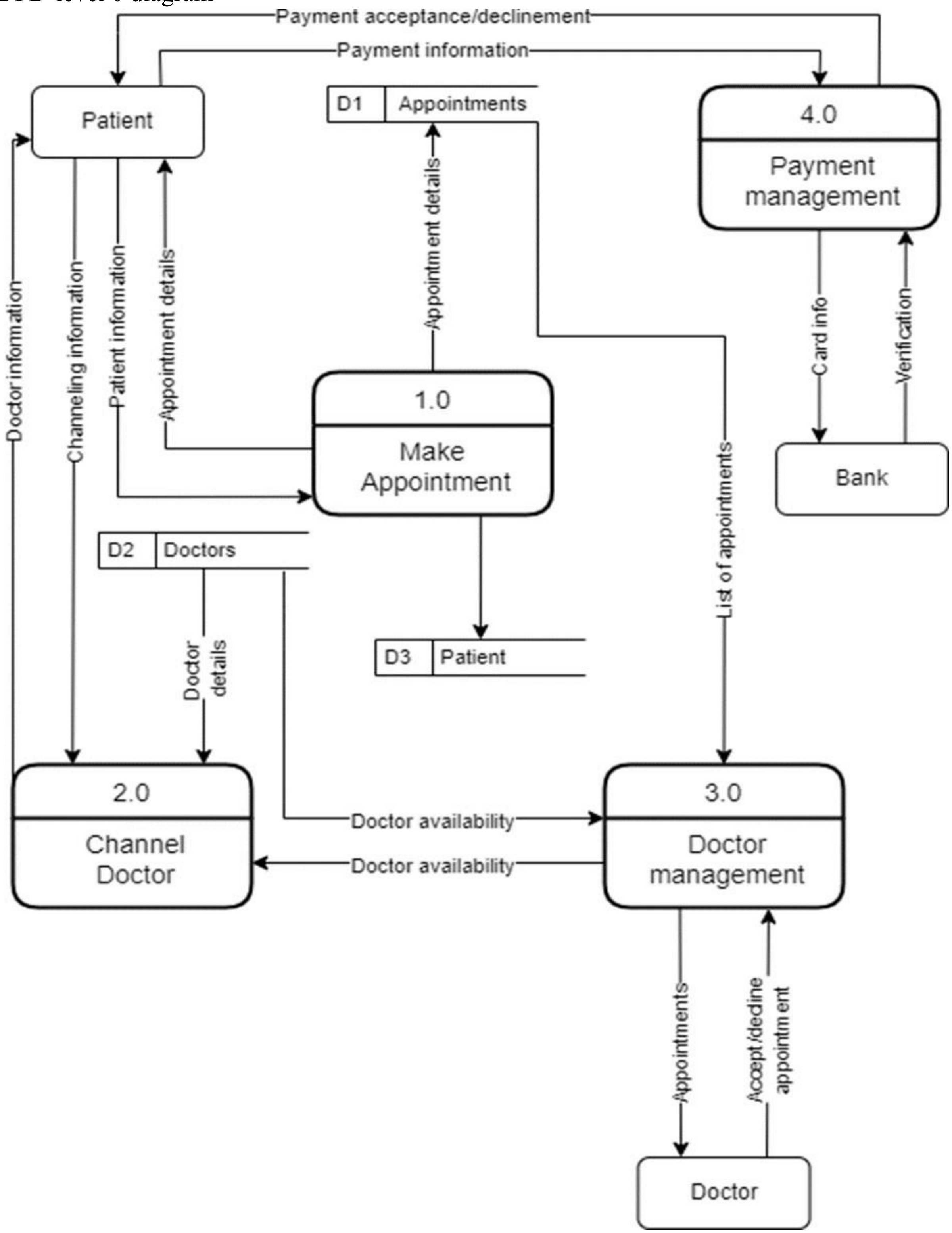
3

Structured system analysis and design method (SSDM)

Context diagram

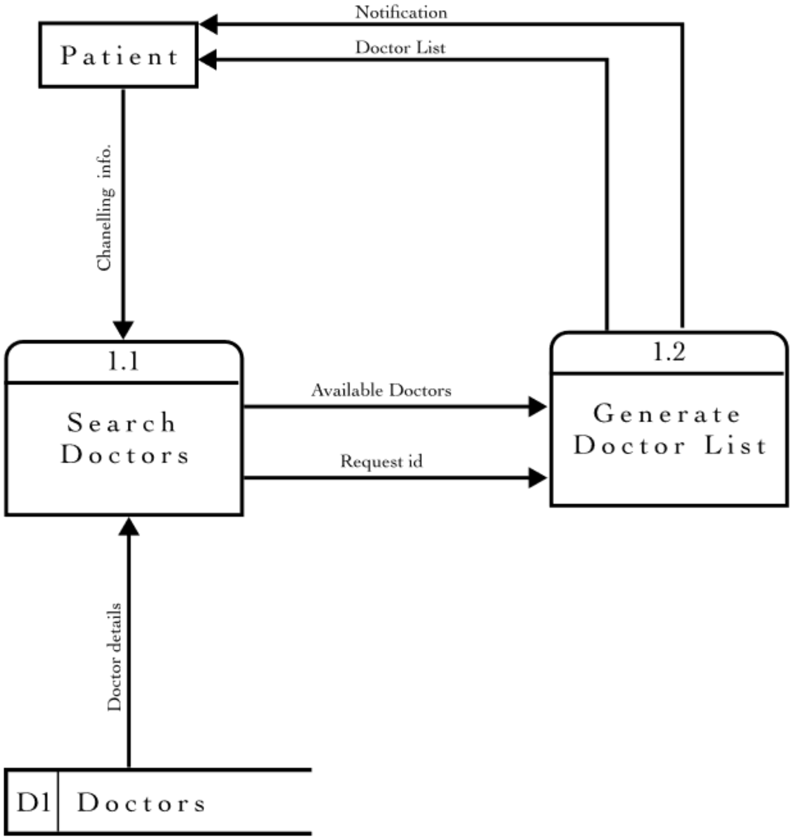


DFD-level 0 diagram



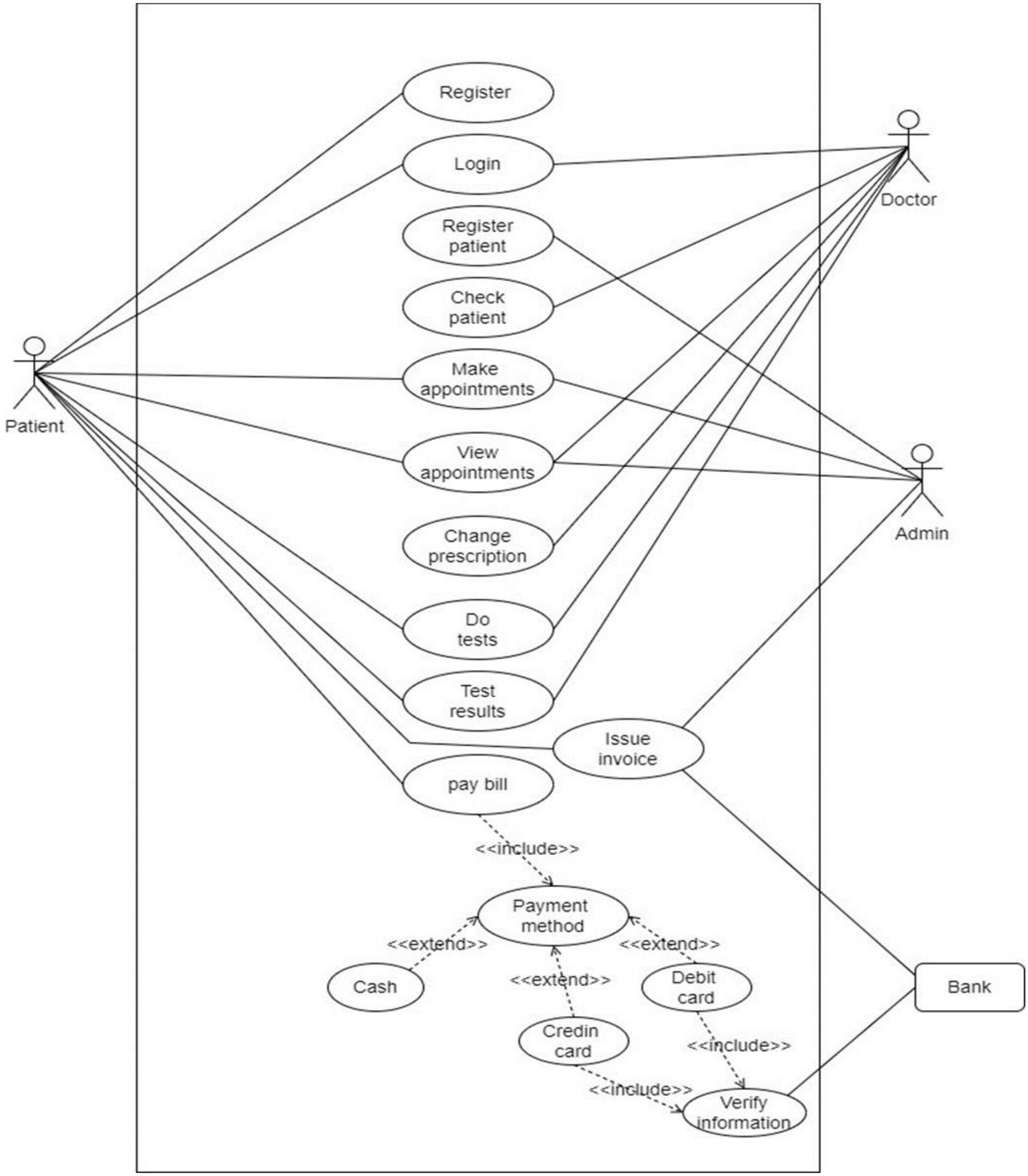
DFD- Level 1

Diagram of Channel Doctor



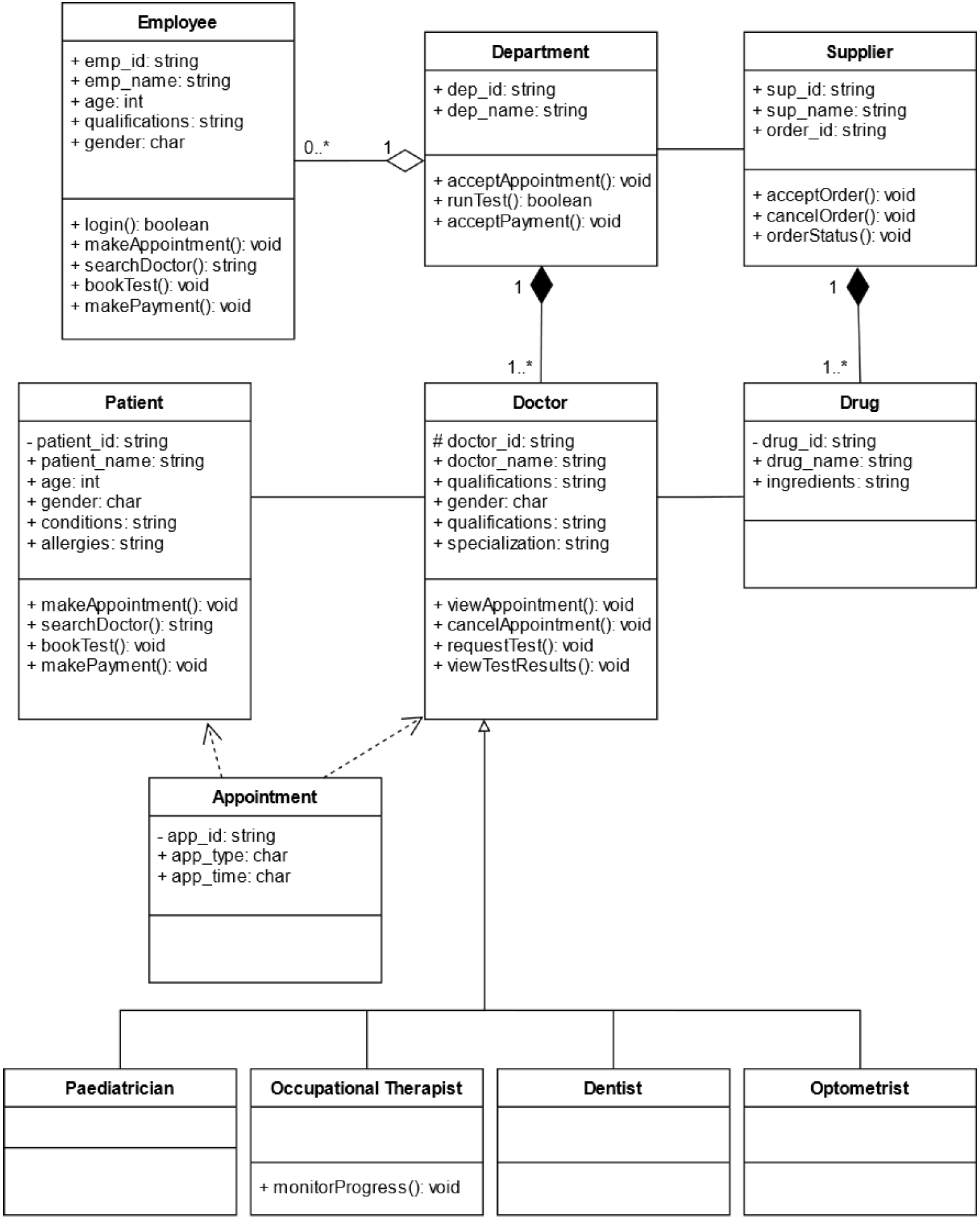
Object Oriented Analysis and Design method (OOADM)

Use case diagram



Class diagram

Class diagram


Employee
+ emp_id: string
+ emp_name: string
+ age: int
+ qualifications: string
+ gender: char
+ login(): boolean
+ makeAppointment(): void
+ searchDoctor(): string
+ bookTest(): void
+ makePayment(): void
0..*
1
Department
+ dep_id: string
+ dep_name: string
+ acceptAppointment(): void
+ runTest(): boolean
+ acceptPayment(): void
Supplier
+ sup_id: string
+ sup_name: string
+ order_id: string
+ acceptOrder(): void
+ cancelOrder(): void
+ orderStatus(): void
1
1..*
1
1..*
Patient
- patient_id: string
+ patient_name: string
+ age: int
+ gender: char
+ conditions: string
+ allergies: string
+ makeAppointment(): void
+ searchDoctor(): string
+ bookTest(): void
+ makePayment(): void
Doctor
# doctor_id: string
+ doctor_name: string
+ qualifications: string
+ gender: char
+ qualifications: string
+ specialization: string
+ viewAppointment(): void
+ cancelAppointment(): void
+ requestTest(): void
+ viewTestResults(): void
Drug
- drug_id: string
+ drug_name: string
+ ingredients: string
Appointment
- app_id: string
+ app_type: char
+ app_time: char
Paediatrician
Occupational Therapist
+ monitorProgress(): void
Dentist
Optometrist