INFO 2413 (A10)
Software Design Descriptions

Jlabs

Version 1.0 approval-pending
Prepared by Team1 : Jason Yu (100301041) , Alex Babenko (100442814), Sabrina Mandla (100423121), Blade Antalik (100436891), Laiba Ali (100403379).

September 26th 2024 Software Design Descriptions for Jlabs

1. Introduction

1.1 Purpose

This document represents the Jlab transitioning into the Development phase, by analyzing the function, non-functional requirements and use cases from the Planning phase new system designs will be produced. Because of doing so our system would have models that elaborate existence and designed types of data, management strategies, data access schemes, and definition of metadata, and description of how to generate reports. These specifications will determine the future implementation of the System, making communication between team members simpler, and making future documentation and analysis of the system simpler.

1.2 Scope

Jlabs is a web-based health monitoring system that clinics use to keep track of information on their patients. This web application will allow patients to register and monitor their own results. It will approve all accounts for users who are registering with the system. The system will also onboard and delete all accounts when hiring new people. Our application will provide a form to create and change exam results. It will provide a feature that lets users select any exam result they want to monitor. When the infrastructure detects any abnormalities in exams, it will notify third party recipients about the issue with an email notification. Users with higher access can choose what they will like to monitor for recipients under care. Monthly and yearly reports give a testing summary of all users abnormal test results. It can also generate health predict reports, outlining various health issues in the next coming years. The system will produce this report from past exams looking back at their history. System can send email notifications and display new information on the web-portal. A search function will be present allowing users to search exam results based on various properties.

The limitations within our system are that people will not be able to book appointments or schedule exams. All exam bookings will be done by something else external to our system. The database can only receive and store their patient exam results. Our system can store these results only after appointments are done and that data is put inside the database. However, certain people will have the ability to modify what is on the exam results. People who do not have accounts with the clinic will not be able to access this application

1.3 Intended audience

This Software Design Descriptions document defines the various architectural designs and details related to implementation that are needed to create a structured and accurate system for Jlabs.The contents of this document are intended to assist project managers, developers, testers, and stakeholders in understanding the elements and their design in the system.

1.4 Reference

"IEEE Standard for Information Technology--Systems Design--Software Design Descriptions," in IEEE Standard for Information Technology--Systems Design--Software Design Descriptions , vol., no..

1.5 Summary

To create a successful implementation of a web-based health monitoring system, Jlabs Is required to design the logical, information, and algorithm viewpoints to understand the framework of the system.

The Logical viewpoint is created to design UML Class Diagrams that show and identify the relationship between different cases. It shows the ways multiple components of a system interact with each other by defining all entities, relationships, and their behaviors. Creating a UML class diagram will provide Jlabs with a blueprint on how the system will and should behave and function. This will help Jlabs plan out the scalability and complexity of the health monitoring system by making it easier to understand the internal framework of the system and the various relationships. Adding new components and implementations in the future will also become easier since this will allow the system to be flexible and adaptable.

The Information viewpoint is created to understand the flow of the system and the relationship between all entities and classes in a diagram. To create the information viewpoint, it is necessary to create detailed database tables that identify relevant columns and rows associated with a specific entity. This helps organize data and information into tables and assigns people, places, and things into their respective tables. This provides Jlabs with a clear view of the different database tables and how and where a table is connected using Primary Keys and Foreign Keys. This ensures accessibility, accuracy, and preciseness while allowing Jlabs to evolve and expand when adding new values into the database tables.

The Algorithm viewpoint is created to provide detailed descriptions regarding the methodology and functionality of the system. This is used to identify what functions are necessary to solve a problem in the system and perform various tasks associated to solve that problem. This will assist Jlabs in creating efficient and reliable algorithms that can be used throughout the system to ensure accurateness and scalability. This will also aid in performance optimization and deter Jlabs from creating mistakes and becoming complex to understand. Algorithm viewpoints will ensure that our system allows the implementation of Artificial Intelligence and machine learning systems in the future.

2. Definitions

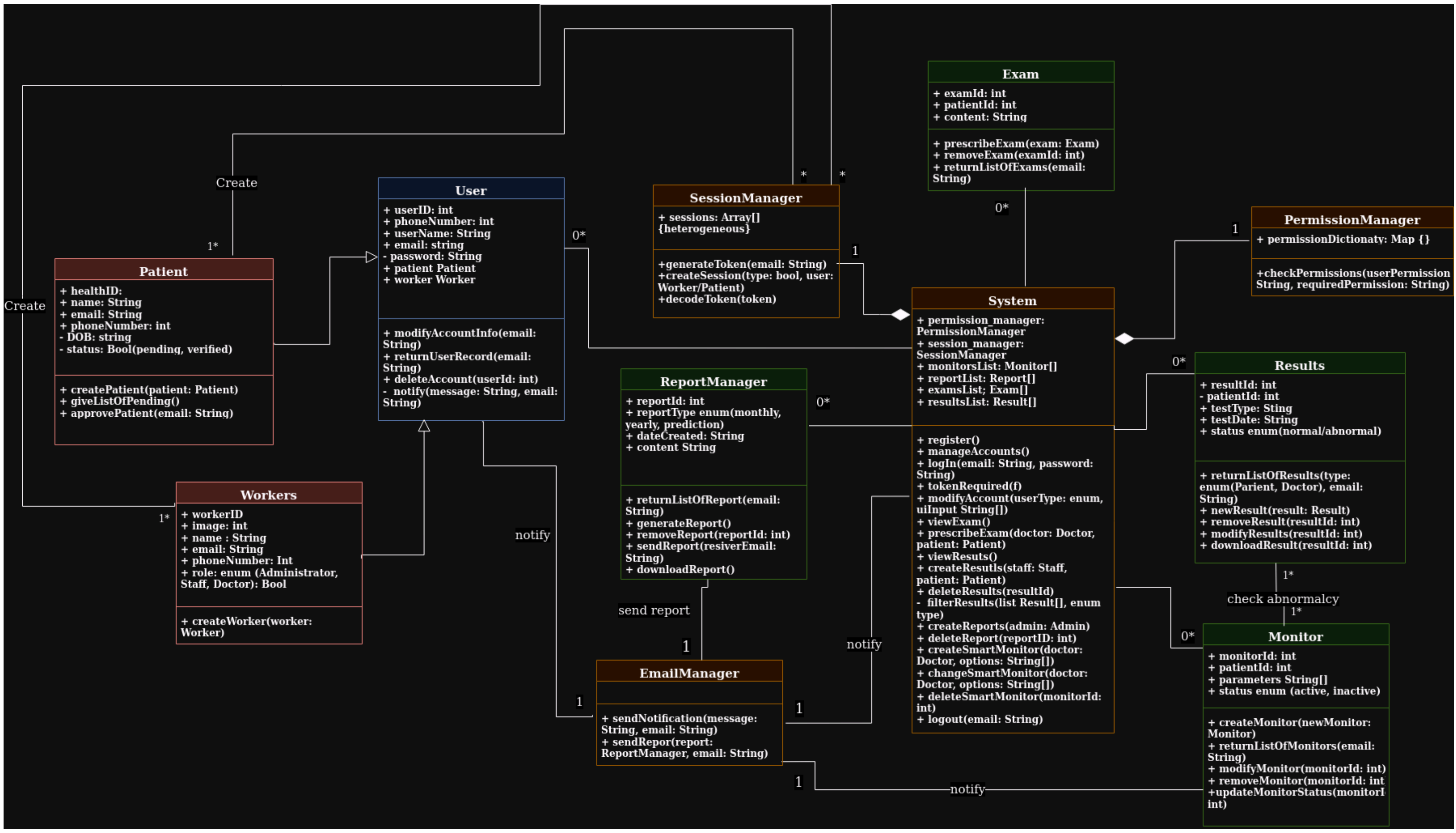
A. SDD (Software Design Document) - a comprehensive document that outlines the architectural design and implementation details of a software system. It includes descriptions of system classes, databases, as well as design description of algorithms
B. Jlabs - is a name of the system described in SDD.
C. Authentication - a function that is verifying user identity using his personal information.
D. Patient - an outside user of the system.
E. Worker - a termine that refers to Doctors, Staff and Administrator that work in the clinic.
F. Staff - a personnel responsible for providing healthcare services in the clinic.
G. Administrator - a personnel responsible for managing the system used by the clinic.
H. Doctor - a physician who diagnoses and treats patients in the clinic.

3. Design viewpoints

3.1 Logical viewpoint

Jlabs as a solution for a small clinic would need to represent its Users in the database and classes. Doctors, Staff and Administrators are represented as “Workers”, the attribute called “UserType” will differentiate them apart and be a key part in the permission management part of the system. Most classes are created with a single responsibility principle in mind, Exam class is only responsible for managing exams, Patient only responsible for patient information. System class is the only class that breaks that rule, it’s a class whose role is to combine and utilize functionality of other classes. For example when a Doctor wants to make a new exam a System class would call an appropriate “prescribeExam()” method that will start the chain of instructions inside of it that will perform needed actions to prescribe exam.

This chain will always consist of the part where the system is using “token_required()” that is checking the token of the user, decodes it, decoded information would include personal information that will be used to check the existence of a user in a database. Followed by it is a “checkPermissions()” that will check the permissions of the user to decide should it perceive or not, and after the initial check “addExam()” will be called that will interact with a database and inform the User back in case of successful insert.


Create
Create
1*
1*
Patient
+ healthID:
+ name: String
+ email: String
+ phoneNumber: int
- DOB: string
- status: Bool(pending, verified)
+ createPatient(patient: Patient)
+ giveListOfPending()
+ approvePatient(email: String)
User
+ userID: int
+ phoneNumber: int
+ userName: String
+ email: string
- password: String
+ patient Patient
+ worker Worker
+ modifyAccountInfo(email: String)
+ returnUserRecord(email: String)
+ deleteAccount(userId: int)
- notify(message: String, email: String)
0*
Workers
+ workerID
+ image: int
+ name : String
+ email: String
+ phoneNumber: Int
+ role: enum (Administrator, Staff, Doctor): Bool
+ createWorker(worker: Worker)
notify
*
*
SessionManager
+ sessions: Array[] {heterogeneous}
+generateToken(email: String)
+createSession(type: bool, user: Worker/Patient)
+decodeToken(token)
1
Exam
+ examId: int
+ patientId: int
+ content: String
+ prescribeExam(exam: Exam)
+ removeExam(examId: int)
+ returnListOfExams(email: String)
0*
PermissionManager
+ permissionDictionaty: Map {}
+checkPermissions(userPermission String, requiredPermission: String)
1
System
+ permission_manager: PermissionManager
+ session_manager: SessionManager
+ monitorsList: Monitor[]
+ reportList: Report[]
+ examsList; Exam[]
+ resultsList: Result[]
+ register()
+ manageAccounts()
+ logIn(email: String, password: String)
+ tokenRequired(f)
+ modifyAccount(userType: enum, uiInput String[])
+ viewExam()
+ prescribeExam(doctor: Doctor, patient: Patient)
+ viewResuts()
+ createResutls(staff: Staff, patient: Patient)
+ deleteResults(resultId)
- filterResults(list Result[], enum type)
+ createReports(admin: Admin)
+ deleteReport(reportID: int)
+ createSmartMonitor(doctor: Doctor, options: String[])
+ changeSmartMonitor(doctor: Doctor, options: String[])
+ deleteSmartMonitor(monitorId: int)
+ logout(email: String)
ReportManager
+ reportId: int
+ reportType enum(monthly, yearly, prediction)
+ dateCreated: String
+ content String
+ returnListOfReport(email: String)
+ generateReport()
+ removeReport(reportId: int)
+ sendReport(resiverEmail: String)
+ downloadReport()
0*
send report
1
EmailManager
+ sendNotification(message: String, email: String)
+ sendRepor(report: ReportManager, email: String)
1
1
notify
1
notify
0*
Results
+ resultId: int
- patientId: int
+ testType: Sting
+ testDate: String
+ status enum(normal/abnormal)
+ returnListOfResults(type: enum(Parient, Doctor), email: String)
+ newResult(result: Result)
+ removeResult(resultId: int)
+ modifyResults(resultId: int)
+ downloadResult(resultId: int)
1*
check abnormalcy
1*
0*
Monitor
+ monitorId: int
+ patientId: int
+ parameters String[]
+ status enum (active, inactive)
+ createMonitor(newMonitor: Monitor)
+ returnListOfMonitors(email: String)
+ modifyMonitor(monitorId: int)
+ removeMonitor(monitorId: int
+updateMonitorStatus(monitorI
int)

3.2 Information viewpoint

Data to be stored in the database will be organized into the following tables. The first row of each table shows the data type of the column in general terms, following rows will show example data of what might appear there. Exact data type will differ based on what is supported by the specific database solution.

Patient:

HealthID(PK)	**Name**	**Email**	**phoneNumber**	**DOB**	**Status**	**DoctorID(FK)**	**Password**
INT (PK)	STRING	STRING	STRING	DATE	BOOL	INT (FK)	String
10031	Bob Ricky	Bobby13@hotmail.com	123-965-4321	05/06/1952	Verified	21002	******
10032	Jenny Kim	Kim.jenn14@yahoo.com	123-524-6352	12/20/1999	Pending	21001	******
10033	Joe Gary	Jojo.g@outlook.com	123-624-3846	10/18/2000	Verified	21001	******
10034	Lisa John	Lisa_00@gamil.com	123-624-3369	02/16/2002	Verified	21002	******
10035	Barry Han	Han.barry50@icloud.com	123-002-5282	08/11/1978	Pending	21001	******
10036	Sean Curry	Seancurry_8@gamil.com	123-945-2361	01/19/2003	Pending	21003	******
10037	Bella Jay	Jay.jayb@hotmail.com	123-777-9056	11/09/2005	Verified	21001	******

Workers Account:

WorkingID (PK)	**Name**	**Email**	**phoneNumber**	**Image**	**Position/userType**	**Password**
INT(PK)	STRING	STRING	INT	BLOB	STRING	STRING
21001	Molly Hue	Molly.Hue@jlabemail.com	789-336-4852	Null	Staff	******
21002	Karen Smith	Karen.Smith@jlabemail.com	789-528-4465	Null	Administrator	******
21003	Jayne Samer	Jayne.Samer@jlabemail.com	789-321-7894	Null	Staff	***********
21004	Lopez Dean	Lopez.Dean@jlabemail.com	789-000-9000	Null	Doctor	*****
21005	Derrick Juan	Derrick.Juan@jlabemail.com	789-987-4356	Null	Staff	*****
21006	Emily Zhang	Emily.Zhang@jlabemail.com	789-176-0986	Null	Staff	*****

21007	Shelly Birch	Shelly.Birch@jlabemail.com	789-923-5981	Null	Staff	*****
21008	Kiana Kir	Kiana.Kir@jlabemail.com	789-008-8000	Null	Doctor	*****
21009	Timothy Collin	Timothy.Collin@jlabemail.com	789-127-7982	Null	Staff	*****
21010	David Wang	David.Wang@jlabemail.com	789-004-4000	Null	Doctor	*****

Exam Table:

ExamID(PK)	Date	healthID(FK)	workingID(FK)	ExamType (FK)
INT (PK)	DATE	INTFK)	INT (FK)	STRING (FK)
33025	10/18/2024	10030	21004	Blood Test
33026	08/01/2024	10031	21010	ECG
33027	11/02/2023	10032	21004	Ultrasound
33028	05/19/2023	10033	21004	X-Ray
33029	08/12/2024	10034	21010	CT scan
33030	03/26/2023	10035	21004	MRI
33031	02/09/2024	10036	21004	Urine Test
33032	10/10/2024	10037	21010	Ultrasound

Exam Types:

ExamType (PK)	Lowerbound	Upperbound	Unit
STRING (FK)	FLOAT	FLOAT	STRING
Blood Test	0.5	1.1	mg/dL
ECG	100	160	mg/dL
Ultrasound	30	140	U/L
X-Ray	7	20	mmHg
CT scan	90	100	%
MRI	7	55	U/L
Urine Test	11	44	ng/mL
Ultrasound	0.0	0.08	%

Exam Results:

ResultID(PK)	ExamID(FK)	Results	date
INT (PK)	INT(FK)	FLOAT	DATE
42057	33025	120	10/20/2024
42058	33026	12.1313	08/10/2024
42059	33027	0.009	11/18/2023
42060	33028	0.0005	05/22/2023
42061	33029	5.5	08/13/2024
42062	33030	52.0	03/28/2023
42063	33031	92.65	02/15/2024
42064	33032	22.50	10/16/2024

Summary Report Table:

SReportID(PK)	workingID(FK)	monthOrYear	Date	TimePeriod
INT (PK)	INT (FK)	STRING/ENUM?	DATE	STRING
5508	21002	Month	10/20/2024	April 2024
5509	21002	Month	11/30/2023	March 2022
5510	21002	Year	06/01/2023	2015

5511	21002	year	08/25/2024	2024
5512	21002	Month	03/28/2023	January 2024
5513	21002	Year	02/28/2024	2023
5514	21002	Year	11/23/2024	2022

Summary Report Entries Table:

SReportID(FK CK)	**HealthID(FK CK)**	**NoOfExams**	**AbnormalExams**
INT(FK)	INT (FK)	INT	INT
5508	21022	5	5
5509	21002	9	4
5510	78002	0	0
5511	41502	2	1
5512	41002	3	0
5513	11002	3	3
5514	21009	1	1

Predict Report Table:

PReportID(PK)	**workingID(FK)**	**PatiendID(FK)**	**Date**
INT (PK)	INT(FK)	INT (FK)	DATE
5508	21002	33026	10/20/2024
5509	21002	33027	11/30/2023
5510	21002	33028	06/01/2023
5511	21002	33029	08/25/2024
5512	21002	33030	03/28/2023
5513	21002	33031	02/28/2024
5514	21002	33032	11/23/2024

Predict Report Entries Table:

PReportID(FK CK)	**ExamType(FK CK)**	**ConcernValue**
INT(FK CK)	STRING (FK CK)	INT
5508	Blood Test	100
5509	Ultrasound	150
5510	Urine Test	25
5511	Urine Test	200
5512	Blood Test	50
5513	Ultrasound	75
5514	Blood Test	120

Smart Monitor:

MonitorID(PK)	workingID(FK)	examType(FK)	status	HealthID(FK)
INT(PK)	INT (FK)	STRING (FK)	BOOL	INT
60001	21004	Blood Test	Sent	1154
60002	21010	Urine Test	Not Sent	4451
60003	21004	CT Scan	Not Sent	1154
60004	21004	CT Scan	Sent	1120
60005	21010	Blood Test	Sent	2150
60006	21004	E33030	Sent	5555
60007	21004	E33031	Not Sent	1234
60008	21010	E33032	Not Sent	4421

3.3 Algorithm viewpoint

There are two types of reports that need to be generated, necessitating two different algorithms. *Italic text* explains the reasoning behind what the algorithm is doing, but does not describe the process itself.

Monthly and Yearly summary:

Technically you could consider these reports distinct, however one algorithm is sufficient to generate either type.

First, prompt the admin for the year or month they want to generate a report for and import all exam results from the specified month or year, store the specified year or month as well.

Next, Import all patients named in exams, or all patients.

For each patient: count total exam results associated with that patient and store the value.

Then count abnormal exam results associated with that patient and store that value.

Retrieve the date and store that value as well. Record which administrator made the report.

At this point the report can be stored in the database by sending the recorded values, displaying the report would use the following instructions:

Display report introduction, this could include the title of the report, what type of report it is, the specified time, and some details of the administrator who made the report.

Then for each patient, display name, number of exams, percentage of abnormal results. Percentage will be calculated at display time using the values stored earlier.

Possibly this could be printed to a text file.

For health prediction report:

Prompt administrator for which patient they wish to generate a report for. Alternatively, simply generate a report for each patient as these are meant to be made all at once at the end of the year.

Either way, for the patient or for each patient:
Import all exam results from patients for the year. Retrieve and record all exam types the patient had that have at least one abnormal result for the year and initialize a concern value for each type, defaulting to some arbitrary number, for example 100. Each recorded exam type for each patient should be stored in its own list or array, and the lists should be sorted by date, with newest results at the end and oldest at the front.

We are not concerned about exam types that only ever gave normal results, only ones that have abnormal results. Each type of exam only has one concern value associated, but every result for that type is stored in the array and will be used to adjust the concern value.

For each exam type the patient had any abnormal result for:
If the last result for that exam type was normal, reduce the concern value for that type by some fixed amount then check previous result for exam type, for each consecutive non-abnormal result reduce concern value further. End this process when an abnormal result is found.

The above instructions will look at issues the patient had but are seemingly resolved, the longer the patient has had normal results, the less likely it is to be a problem again.

For any exam type with only one exam result, leave concern value at default.

For exam types with the last result being abnormal and more than one exam, initialize a multiplier value as 1. Then, check previous test results and if it is also abnormal increase the concern value by some number multiplied with the multiplier value. If instead a normal result is reached, divide the concern multiplier by two. Then repeat the process until all results are iterated through.

The more consecutive abnormal results the patient has had, the more confident we can be that this will be a continuing health problem for the patient going forward. Unlike before, we do not stop as soon as we reach a different result type, this way a single normal result does not cause us to ignore long records of bad results. However we do not want an abnormal result to carry heavy weight if many normal results followed it before another abnormal result. To solve this, a multiplier value is used, and lowered every time a normal result is found.

At this point the report information can be stored in our database by sending each exam type and its concern value, as well as the date of the report, who generated it, and which patient it is for.

To display the report do as follows:

Display title of report, likely something like “Prediction report for Patient”.

Display basic patient info and date the report was made.

Sort exam types by their concern values.

Then for each type display the exam type, its concern value, and a message based on the value.

If the value is greater than some higher range, say 200, predict a very high risk of serious health issues.

For values between 150-199, predict high risk of serious health issues. For values 101-149, predict elevated risk of serious health issues.

For a default value of 100, predict possible health issues.

For ranges 75-99, predict possible re-emergence of health issue and recommend continued monitoring and action

For ranges below 75, predict high probability that the health issue is resolved.

4. Changes to non-functional requirements

In the SRS relating to the project, too many non-functional requirements were listed, some of which are not actually feasible. The following three requirements are what we will focus on:

A. Search Operations: ≤ 5 seconds for up to 500 records 95% of the time.

B. Report Generation: ≤ 30 seconds for summary report with less than 500 entries; prediction reports ≤ 10 seconds 99% of the time.

C. Users must be able to reach the intended page by clicking 5 or fewer links.

5. Appendix A: Glossary