Reviewed by:
Approved by:

Revision / Document History

Ver.	Date	Changed by	Modifications
1.0	1/8/2023	Bakkesh S Subedhar	Initial version of document, Database design is included in this document, review changes implemented
1.1	2/8/2023	Bakkesh S Subedhar	1. Entity Relationship Diagram introduced in Sec 9.1 2. Sec 5: Added Guidelines on Software Design Characteristics 3. Sec 5 (4) Efficiency has been changed 4. Sec 8 the fault tolerant has been changed for clarity 5. Sec 15 Black box and white Box description has been removed 6. Sec 16 Traceability table has been removed and reference has been given to Requirements guidelines in Sec 16.
1.2	3/8/2023	Bakkesh S Subedhar	Section 5: Elaborated– database design considerations, added plan for upgrades ,backup and recover planning section 6: added internal and external interfaces to give more clarity
1.3	4/8/2023	Bakkesh S Subedhar	Enhanced section on Interface design to include component design details Included reference to DAR template for alternate design solutions
1.4	5/8/2023	Bakkesh S Subedhar	Modified the template according to CORE-TM(F) training needs.

Note: Present content to be deleted and revision history to be updated with project's document history, when used in Projects

List of Abbreviations

DFD	**D**ata **F**low **D**iagram
ER	**E**ntity **R**elationship
FHD	**F**unction **H**ierarchy **D**iagram
HLD	**H**igh **L**evel **D**esign
LLD	**L**ow **L**evel **D**esign
GUI	**G**raphical **U**ser **I**nterface
IEEE	**I**nstitute of **E**lectrical and **E**lectronic **E**ngineers
S/W	**S**oft**w**are
SDL	**S**pecification **D**escription **L**anguage
StrD	**Str**ucture**d**

TABLE OF CONTENTS

1. Introduction

This project aims to develop an cashless based service, which enables an automated system for patient admission, discharge and transfer process (Ward Transfer) within the HealthSure hospital.

2. Design Scope

The scope of the project is limited to patient admission, transfer (Ward transfer) and discharge. This process enables the patient to book an available ward for admission / transfer if required, and enables the hospital admin to have complete supervision on the patients requests.

3. Design Methodology

Object Oriented Analysis and Design (OOAD) methodology has been used for breaking down the specification into functionally independent units.

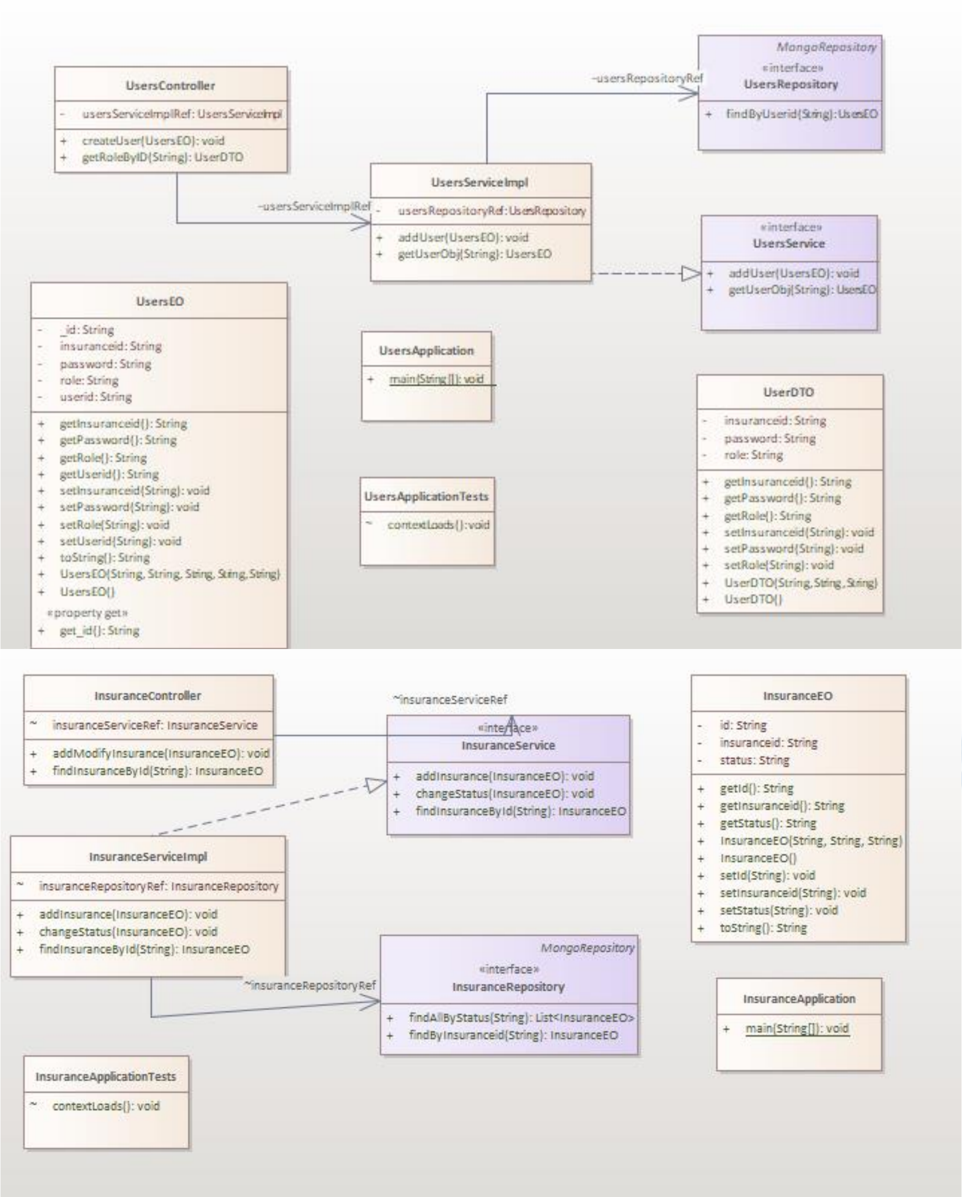
4. Design Notations

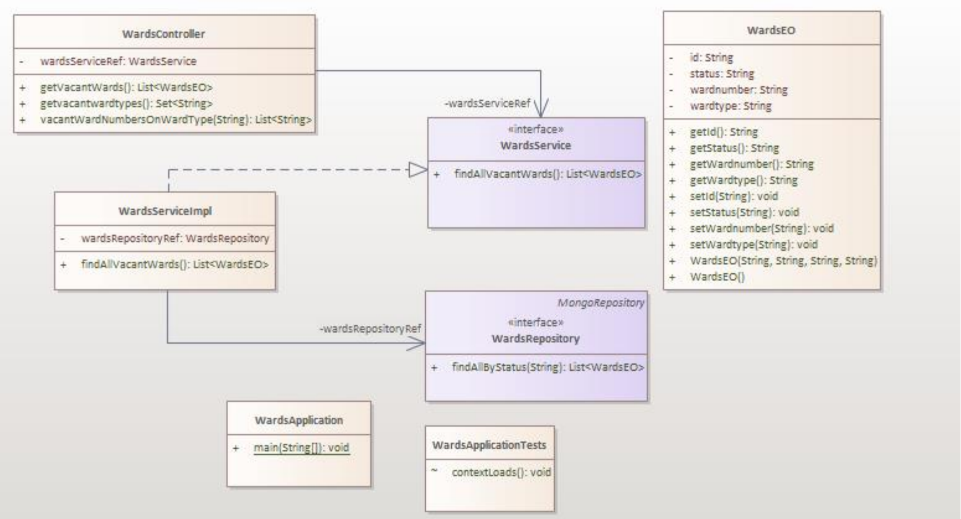
The naming conventions followed conform to Unified Modelling Language (UML) as Object Oriented Analysis and Design (OOAD) is followed.

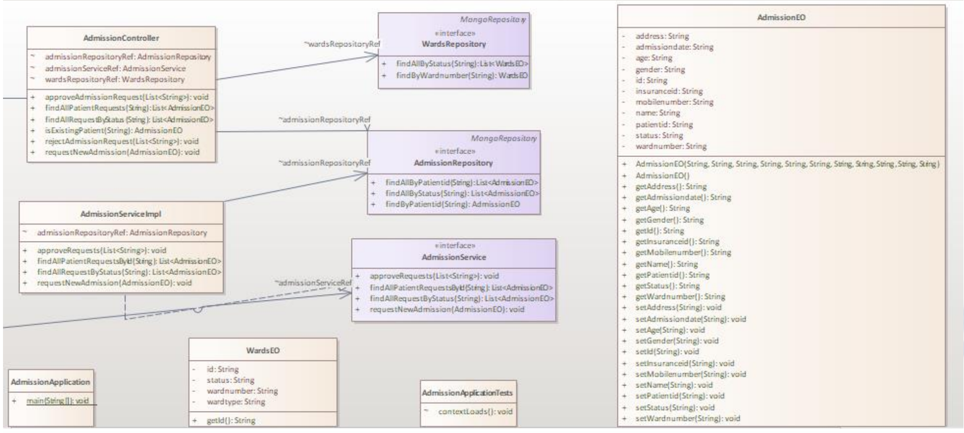
5. Design Considerations

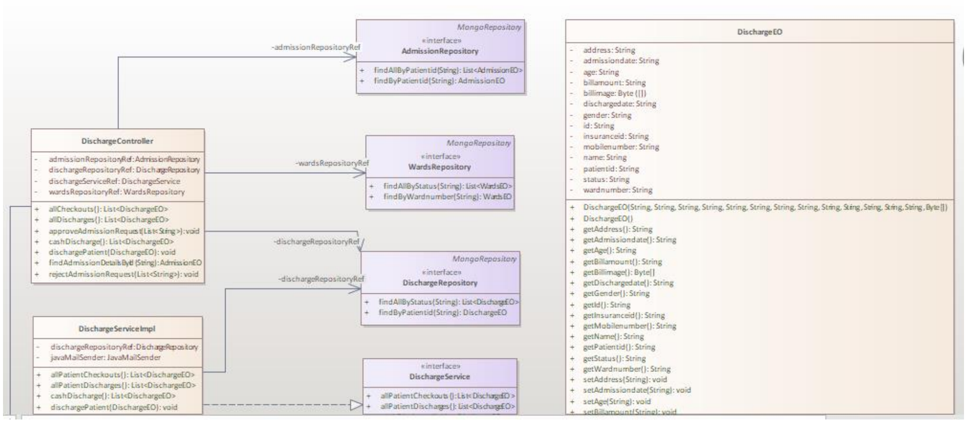
Not Applicable.

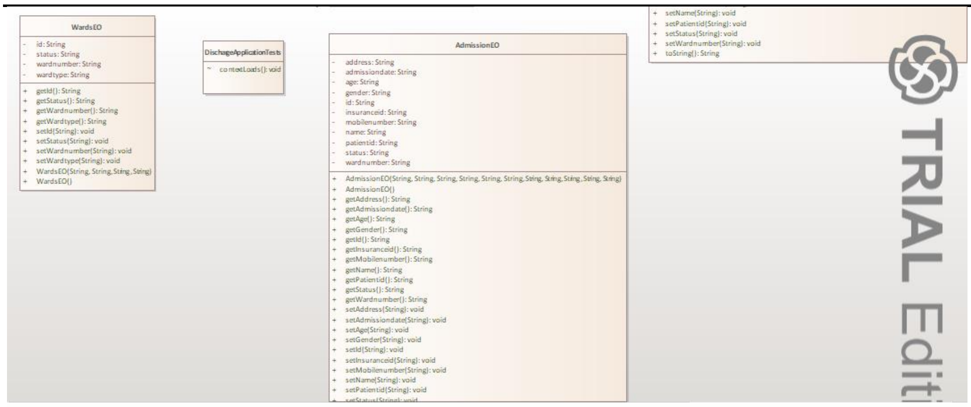
6. Design Overview

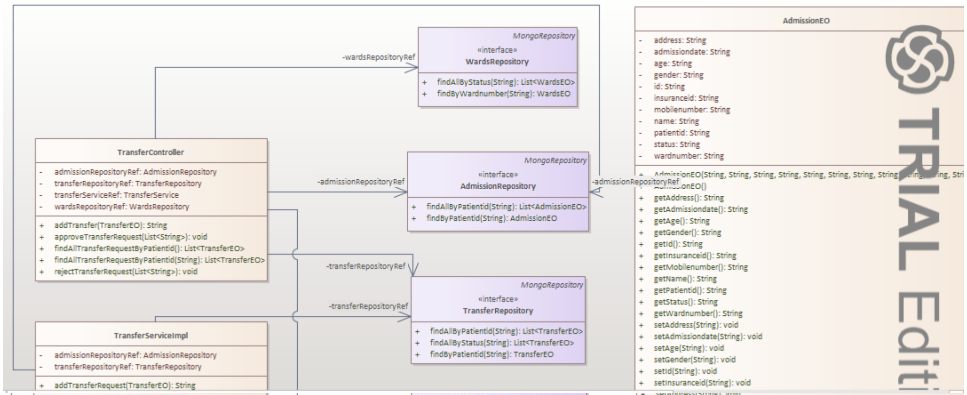

UsersController
- usersServiceImplRef: UsersServiceImpl
+ createUser(UsersEO): void
+ getRoleByID(String): UserDTO
~usersRepositoryRef
MongoRepository
«interface»
UsersRepository
+ findByUserid(String): UsersEO
~usersServiceImplRef
UsersServiceImpl
- usersRepositoryRef: UsersRepository
+ addUser(UsersEO): void
+ getUserObj(String): UsersEO
«interface»
UsersService
+ addUser(UsersEO): void
+ getUserObj(String): UsersEO
UsersEO
- _id: String
- insuranceid: String
- password: String
- role: String
- userid: String
+ getInsuranceid(): String
+ getPassword(): String
+ getRole(): String
+ getUserid(): String
+ setInsuranceid(String): void
+ setPassword(String): void
+ setRole(String): void
+ setUserid(String): void
+ toString(): String
+ UsersEO(String, String, String, String, String)
+ UsersEO()
«property get»
+ get_id(): String
UsersApplication
+ main(String[]): void
UsersApplicationTests
~ contextLoads():void
UserDTO
- insuranceid: String
- password: String
- role: String
+ getInsuranceid(): String
+ getPassword(): String
+ getRole(): String
+ setInsuranceid(String): void
+ setPassword(String): void
+ setRole(String): void
+ UserDTO(String, String, String)
+ UserDTO()
InsuranceController
~ insuranceServiceRef: InsuranceService
+ addModifyInsurance(InsuranceEO): void
+ findInsuranceById(String): InsuranceEO
~insuranceServiceRef
«interface»
InsuranceService
+ addInsurance(InsuranceEO): void
+ changeStatus(InsuranceEO): void
+ findInsuranceById(String): InsuranceEO
InsuranceEO
- id: String
- insuranceid: String
- status: String
+ getId(): String
+ getInsuranceid(): String
+ getStatus(): String
+ InsuranceEO(String, String, String)
+ InsuranceEO()
+ setId(String): void
+ setInsuranceid(String): void
+ setStatus(String): void
+ toString(): String
InsuranceServiceImpl
~ insuranceRepositoryRef: InsuranceRepository
+ addInsurance(InsuranceEO): void
+ changeStatus(InsuranceEO): void
+ findInsuranceById(String): InsuranceEO
~insuranceRepositoryRef
MongoRepository
«interface»
InsuranceRepository
+ findAllByStatus(String): List<InsuranceEO>
+ findByInsuranceid(String): InsuranceEO
InsuranceApplication
+ main(String[]): void
InsuranceApplicationTests
~ contextLoads(): void

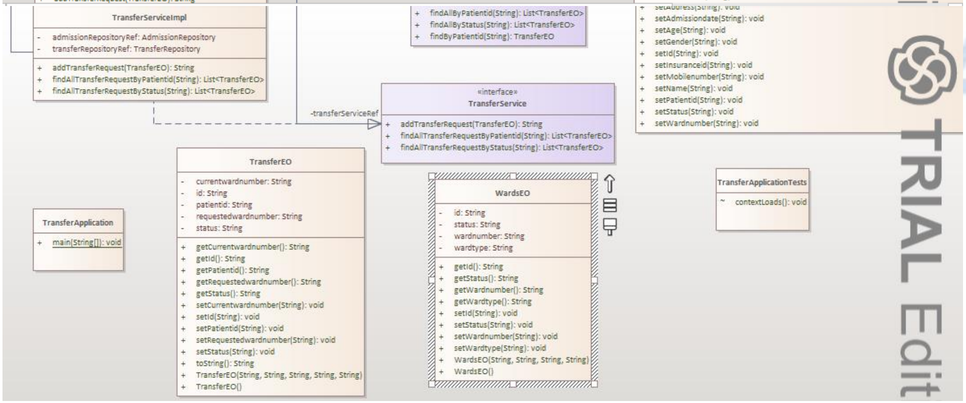

WardsController
- wardsServiceRef: WardsService
+ getVacantWards(): List<WardsEO>
+ getvacantwardtypes(): Set<String>
+ vacantWardNumbersOnWardType(String): List<String>
-wardsServiceRef
«interface»
WardsService
+ findAllVacantWards(): List<WardsEO>
WardsServiceImpl
- wardsRepositoryRef: WardsRepository
+ findAllVacantWards(): List<WardsEO>
-wardsRepositoryRef
MongoRepository
«interface»
WardsRepository
+ findAllByStatus(String): List<WardsEO>
WardsEO
- id: String
- status: String
- wardnumber: String
- wardtype: String
+ getId(): String
+ getStatus(): String
+ getWardnumber(): String
+ getWardtype(): String
+ setId(String): void
+ setStatus(String): void
+ setWardnumber(String): void
+ setWardtype(String): void
+ WardsEO(String, String, String, String)
+ WardsEO()
WardsApplication
+ main(String[]): void
WardsApplicationTests
~ contextLoads(): void


AdmissionController
~ admissionRepositoryRef: AdmissionRepository
~ admissionServiceRef: AdmissionService
~ wardsRepositoryRef: WardsRepository
+ approveAdmissionRequest(List<String>): void
+ findAllPatientRequests(String): List<AdmissionEO>
+ findAllRequestByStatus (String): List<AdmissionEO>
+ isExistingPatient(String): AdmissionEO
+ rejectAdmissionRequest(List<String>): void
+ requestNewAdmission(AdmissionEO): void
~wardsRepositoryRef
MongoRepository
«interface»
WardsRepository
+ findAllByStatus(String): List<WardsEO>
+ findByWardnumber(String): WardsEO
~admissionRepositoryRef
~admissionRepositoryRef
MongoRepository
«interface»
AdmissionRepository
+ findAllByPatientid(String): List<AdmissionEO>
+ findAllByStatus(String): List<AdmissionEO>
+ findByPatientid(String): AdmissionEO
AdmissionServiceImpl
~ admissionRepositoryRef: AdmissionRepository
+ approveRequests(List<String>): void
+ findAllPatientRequestsById(String): List<AdmissionEO>
+ findAllRequestByStatus(String): List<AdmissionEO>
+ requestNewAdmission(AdmissionEO): void
~admissionServiceRef
«interface»
AdmissionService
+ approveRequests(List<String>): void
+ findAllPatientRequestsById(String): List<AdmissionEO>
+ findAllRequestByStatus(String): List<AdmissionEO>
+ requestNewAdmission(AdmissionEO): void
AdmissionEO
- address: String
- admissiondate: String
- age: String
- gender: String
- id: String
- insuranceid: String
- mobilenumber: String
- name: String
- patientid: String
- status: String
- wardnumber: String
+ AdmissionEO(String, String, String, String, String, String, String, String, String, String, String)
+ AdmissionEO()
+ getAddress(): String
+ getAdmissiondate(): String
+ getAge(): String
+ getGender(): String
+ getId(): String
+ getInsuranceid(): String
+ getMobilenumber(): String
+ getName(): String
+ getPatientid(): String
+ getStatus(): String
+ getWardnumber(): String
+ setAddress(String): void
+ setAdmissiondate(String): void
+ setAge(String): void
+ setGender(String): void
+ setId(String): void
+ setInsuranceid(String): void
+ setMobilenumber(String): void
+ setName(String): void
+ setPatientid(String): void
+ setStatus(String): void
+ setWardnumber(String): void
WardsEO
- id: String
- status: String
- wardnumber: String
- wardtype: String
+ getId(): String
AdmissionApplication
+ main(String[]): void
AdmissionApplicationTests
~ contextLoads(): void


MongoRepository
«interface»
AdmissionRepository
+ findAllByPatientid(String): List<AdmissionEO>
+ findByPatientid(String): AdmissionEO
-admissionRepositoryRef
DischargeController
- admissionRepositoryRef: AdmissionRepository
- dischargeRepositoryRef: DischargeRepository
- dischargeServiceRef: DischargeService
- wardsRepositoryRef: WardsRepository
+ allCheckouts(): List<DischargeEO>
+ allDischarges(): List<DischargeEO>
+ approveAdmissionRequest(List<String>): void
+ cashDischarge(): List<DischargeEO>
+ dischargePatient(DischargeEO): void
+ findAdmissionDetailsById (String): AdmissionEO
+ rejectAdmissionRequest(List<String>): void
-wardsRepositoryRef
MongoRepository
«interface»
WardsRepository
+ findAllByStatus(String): List<WardsEO>
+ findByWardnumber(String): WardsEO
-dischargeRepositoryRef
-dischargeRepositoryRef
MongoRepository
«interface»
DischargeRepository
+ findAllByStatus(String): List<DischargeEO>
+ findByPatientid(String): DischargeEO
DischargeServiceImpl
- dischargeRepositoryRef: DischargeRepository
- javaMailSender: JavaMailSender
+ allPatientCheckouts(): List<DischargeEO>
+ allPatientDischarges(): List<DischargeEO>
+ cashDischarge(): List<DischargeEO>
+ dischargePatient(DischargeEO): void
«interface»
DischargeService
+ allPatientCheckouts (): List<DischargeEO>
+ allPatientDischarges(): List<DischargeEO>
DischargeEO
- address: String
- admissiondate: String
- age: String
- billamount: String
- billimage: Byte ([])
- dischargedate: String
- gender: String
- id: String
- insuranceid: String
- mobilenumber: String
- name: String
- patientid: String
- status: String
- wardnumber: String
+ DischargeEO(String, String, String, String, String, String, String, String, String, String, String, String, String, Byte[])
+ DischargeEO()
+ getAddress(): String
+ getAdmissiondate(): String
+ getAge(): String
+ getBillamount(): String
+ getBillimage(): Byte[]
+ getDischargedate(): String
+ getGender(): String
+ getId(): String
+ getInsuranceid(): String
+ getMobilenumber(): String
+ getName(): String
+ getPatientid(): String
+ getStatus(): String
+ getWardnumber(): String
+ setAddress(String): void
+ setAdmissiondate(String): void
+ setAge(String): void
+ setBillamount(String): void


WardsEO
- id: String
- status: String
- wardnumber: String
- wardtype: String
+ getId(): String
+ getStatus(): String
+ getWardnumber(): String
+ getWardtype(): String
+ setId(String): void
+ setStatus(String): void
+ setWardnumber(String): void
+ setWardtype(String): void
+ WardsEO(String, String, String, String)
+ WardsEO()
DischageApplicationTests
~ contextLoads(): void
AdmissionEO
- address: String
- admissiondate: String
- age: String
- gender: String
- id: String
- insuranceid: String
- mobilenumber: String
- name: String
- patientid: String
- status: String
- wardnumber: String
+ AdmissionEO(String, String, String, String, String, String, String, String, String, String, String)
+ AdmissionEO()
+ getAddress(): String
+ getAdmissiondate(): String
+ getAge(): String
+ getGender(): String
+ getId(): String
+ getInsuranceid(): String
+ getMobilenumber(): String
+ getName(): String
+ getPatientid(): String
+ getStatus(): String
+ getWardnumber(): String
+ setAddress(String): void
+ setAdmissiondate(String): void
+ setAge(String): void
+ setGender(String): void
+ setId(String): void
+ setInsuranceid(String): void
+ setMobilenumber(String): void
+ setName(String): void
+ setPatientid(String): void
+ setName(String): void
+ setPatientid(String): void
+ setStatus(String): void
+ setWardnumber(String): void
+ toString(): String
TRIAL Edition


MongoRepository
«interface»
WardsRepository
+ findAllByStatus(String): List<WardsEO>
+ findByWardnumber(String): WardsEO
-wardsRepositoryRef
TransferController
- admissionRepositoryRef: AdmissionRepository
- transferRepositoryRef: TransferRepository
- transferServiceRef: TransferService
- wardsRepositoryRef: WardsRepository
+ addTransfer(TransferEO): String
+ approveTransferRequest(List<String>): void
+ findAllTransferRequestByPatientid(): List<TransferEO>
+ findAllTransferRequestByPatientid(String): List<TransferEO>
+ rejectTransferRequest(List<String>): void
-admissionRepositoryRef
MongoRepository
«interface»
AdmissionRepository
+ findAllByPatientid(String): List<AdmissionEO>
+ findByPatientid(String): AdmissionEO
-admissionRepositoryRef
-transferRepositoryRef
-transferRepositoryRef
MongoRepository
«interface»
TransferRepository
+ findAllByPatientid(String): List<TransferEO>
+ findAllByStatus(String): List<TransferEO>
+ findByPatientid(String): TransferEO
TransferServiceImpl
- admissionRepositoryRef: AdmissionRepository
- transferRepositoryRef: TransferRepository
+ addTransferRequest(TransferEO): String
AdmissionEO
- address: String
- admissiondate: String
- age: String
- gender: String
- id: String
- insuranceid: String
- mobilenumber: String
- name: String
- patientid: String
- status: String
- wardnumber: String
+ AdmissionEO(String, String, String, String, String, String, String, String, String, String, String)
+ AdmissionEO()
+ getAddress(): String
+ getAdmissiondate(): String
+ getAge(): String
+ getGender(): String
+ getId(): String
+ getInsuranceid(): String
+ getMobilenumber(): String
+ getName(): String
+ getPatientid(): String
+ getStatus(): String
+ getWardnumber(): String
+ setAddress(String): void
+ setAdmissiondate(String): void
+ setAge(String): void
+ setGender(String): void
+ setId(String): void
+ setInsuranceid(String): void
TRIAL Edition


+ findAllByPatientid(String): List<TransferEO>
+ findAllByStatus(String): List<TransferEO>
+ findByPatientid(String): TransferEO
TransferServiceImpl
- admissionRepositoryRef: AdmissionRepository
- transferRepositoryRef: TransferRepository
+ addTransferRequest(TransferEO): String
+ findAllTransferRequestByPatientid(String): List<TransferEO>
+ findAllTransferRequestByStatus(String): List<TransferEO>
-transferServiceRef
«interface»
TransferService
+ addTransferRequest(TransferEO): String
+ findAllTransferRequestByPatientid(String): List<TransferEO>
+ findAllTransferRequestByStatus(String): List<TransferEO>
TransferEO
- currentwardnumber: String
- id: String
- patientid: String
- requestedwardnumber: String
- status: String
+ getCurrentwardnumber(): String
+ getId(): String
+ getPatientid(): String
+ getRequestedwardnumber(): String
+ getStatus(): String
+ setCurrentwardnumber(String): void
+ setId(String): void
+ setPatientid(String): void
+ setRequestedwardnumber(String): void
+ setStatus(String): void
+ toString(): String
+ TransferEO(String, String, String, String, String)
+ TransferEO()
TransferApplication
+ main(String[]): void
WardsEO
- id: String
- status: String
- wardnumber: String
- wardtype: String
+ getId(): String
+ getStatus(): String
+ getWardnumber(): String
+ getWardtype(): String
+ setId(String): void
+ setStatus(String): void
+ setWardnumber(String): void
+ setWardtype(String): void
+ WardsEO(String, String, String, String)
+ WardsEO()
TransferApplicationTests
~ contextLoads(): void
+ setAdmissiondate(String): void
+ setAge(String): void
+ setGender(String): void
+ setId(String): void
+ setInsuranceid(String): void
+ setMobilenumber(String): void
+ setName(String): void
+ setPatientid(String): void
+ setStatus(String): void
+ setWardnumber(String): void
TRIAL Edit

7. Decomposition

DH-1-1 User Master

Inputs:	Sets the log in details.
Outputs:	Gets the log in details.
Scope:	Specific

This class is a model class for the database Users Collection.

DH-1-2 Admission Master

Inputs:	Sets the Admission details.
Outputs:	Gets the Admission details.
Scope:	Specific

This class is a model class for the database Admission Collection.

DH-1-3 Discharge Master

Inputs:	Sets the Discharge details.
Outputs:	Gets the Discharge details.
Scope:	Specific

This class is a model class for the database Admission Collection.

DH-1-4 Wards Master

Inputs:	Sets the wards details.
Outputs:	Gets the wards details.
Scope:	Specific

This class is a model class for the database Wards Collection

DH-1-5 Insurance Master

Inputs:	Sets the Insurance details.
Outputs:	Gets the Insurance details.
Scope:	Specific

This class is a model class for the database Insurance Collection.

DH-1-6 Transfer Master

Inputs:	Sets the Wards details.
Outputs:	Gets the ward Transfer details.

Scope: Specific

This class is a model class for the database Transfer Collection.

DH-1-8 UserServiceImpl

Inputs: The details to be updated, selected or deleted.
Outputs: The result of selection, updating or deletion.
Scope: Specific

This class implements the UserDAO interface.

DH-1-9 WardsServiceImpl

Inputs: The details to be updated, selected or deleted.
Outputs: The result of selection, updating or deletion.
Scope: Specific

This class implements the WardsService interface.

DH-1-10 TransferServiceImpl

Inputs: The details to be updated, selected or deleted.
Outputs: The result of selection, updating or deletion.
Scope: Generic

This class implements the TransferService interface.

DH-1-11 AdmissionServiceImpl

Inputs: The details to be updated, selected or deleted.
Outputs: The result of selection, updating or deletion.
Scope: Specific

This class implements the AdmissionService interface.

DH-1-12 DischargeServiceImpl

Inputs: The details to be updated, selected or deleted.
Outputs: The result of selection, updating or deletion.
Scope: Specific

This class implements the DischargeService interface.

DH-1-13 InsuranceServiceImpl

Inputs:	The details to be updated, selected or deleted.
Outputs:	The result of selection, updating or deletion.
Scope:	Specific

This class implements the InsuranceService interface.

8. Interface Design

8.1 User Interface

NA

9.2 Data structure (data types, arrays, and structures)

Not applicable.

10. Reusability

- Authentication
- Validation
- ResourceManager

11. Design Alternatives

NA

12. Design Feasibility

We have used the OOAD approach in this project. This methodology has been chosen based on our analogy of the user requirements, feasibility study and based on the experience of the co-ordinators. It has been seen that several other project groups developing similar projects have chosen the same methodology.

The OOAD assures properties such as reusability, modularity, efficiency.

13. Additional Hardware and Software required

This requirement is based on the future stages of development. Therefore as of now this is not applicable

14. Testing Strategy

The various stages of testing to be followed for our application includes white unit and integration testing.
We will carry out all such testing in a simulated environment only.

15. Traceability Matrix

As per the requirements-HLD tagging shown in the document “Requirement_Traceability.xls” each of the requirements has been mapped to the appropriate classes. Both the requirements and classes have been tagged according to the tag standards of RBIN.

16. References

List of all external sources of information referenced in this document.

Sl. No.	Description	Date	Vers.	Location
1.	Software Requirements Specification Document	05/08/2023	1.0	*SRS.doc*
2.	Low Level Design Document	06/08/2023	1.0	
3.				

Description, date, and version shall uniquely identify the information source, and the location shall specify where it is to be found.