

Software Application for Brigham & Women's Hospital





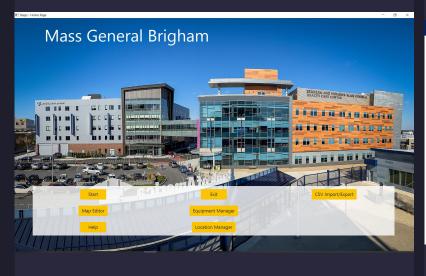




BEFORE & AFTER: APPLICATION

HOME PAGE

BEFORE



AFTER



MAP EDITOR

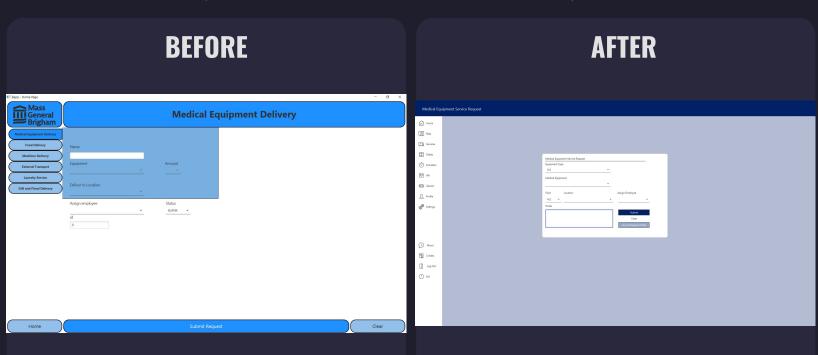
BEFORE



AFTER



MEDICAL EQUIPMENT SERVICE REQUEST



Goal of Team and My Role

Team

- Team of 10 people applying Agile Scrum methodology.
- Developed a desktop application for the hospital employees to access and use.
- Application focuses on creating and carrying out Service Requests.
- Database tables are maintained and implemented in an easy-to-use graphical interface
- Conducted daily scrums, sprint plannings and sprint reflections.

My Role

- Designed and developed back-end system including database management system and restore/backup subsystem.
- Implemented embedded and client-server database using Apache Derby and remote cloud-based database using MongoDB.
- As Assistant Lead Software Engineer, coordinated the other software engineers, particularly the back-end team, and performed code reviews through Git workflows.

118

JAVA CLASSES

3

SUBSYSTEMS

5

INTERFACES

15,424

LINES OF JAVA CODE

Diagrams

First Iteration Vs. Last Iteration UML Diagrams

First Last

First Iteration Vs. Last Iteration ERDs

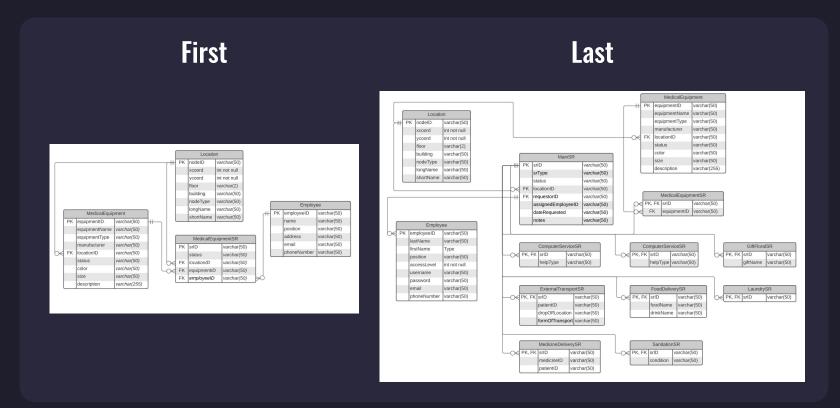
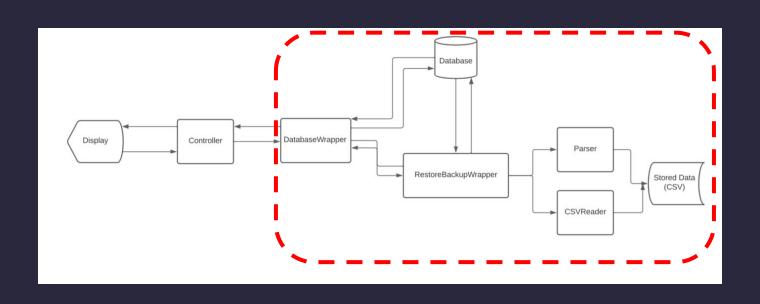
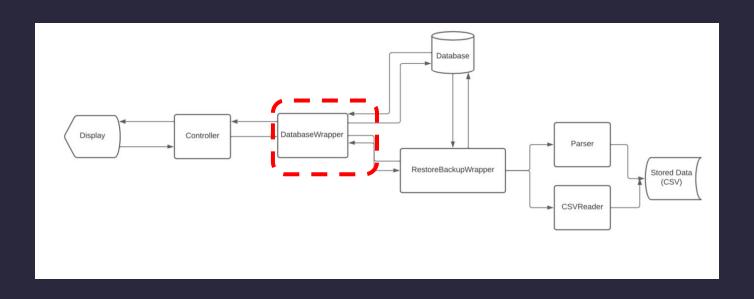


Diagram of Systems and Subsystems



DatabaseWrapper

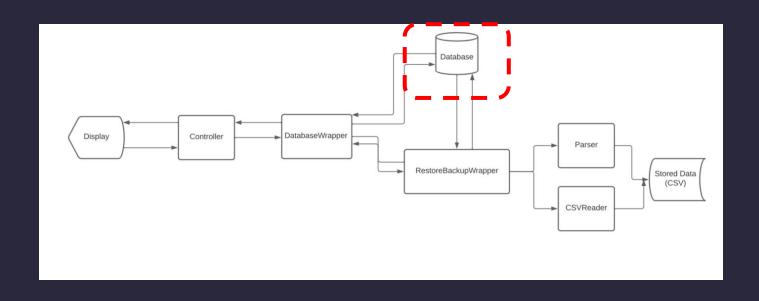


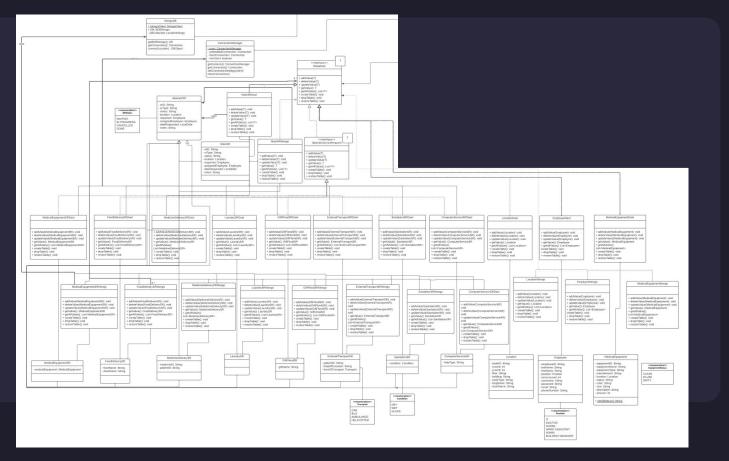
DatabaseWrapper

- Facade pattern
- Front-facing interface
- Masks complex database code and is what front-end uses to manipulate database
- Singleton pattern implementation

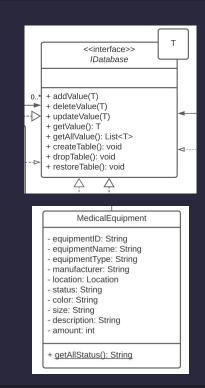
DatabaseWrapper DatabaseWrapperInstance: DatabaseWrapper LocationDao: IDatabase<Location> LocationDerby: IDatabase<Location> LocationMongo: IDatabase<Location> connectionManager: ConnectionManager restoreBackupWrapper: RestoreBackupWrapper + getInstance(): DatabaseWrapper + engageEmbedded() + engageClient() + engageRemote() + initEmbedded() + initClient() + setRestoreBackupDao() + addSR(AbstractSR) + addLocation(Location) + addEmployee(Employee) + addMedicalEquipment(MedicalEquipment) + deleteSR(String id) + deleteLocation(String id) + deleteEmployee(Employee id) + deleteMedicalEquipment(String id) + updateSR(AbstractSR) + updateLocation(Location) + updateEmployee(Employee) + updateMedicalEquipment(MedicalEquipment) + getValueSR(): AbstractSR + getValueLocation(): Location + getValueEmployee(): Employee + getValueMedicalEquipment(): MedicalEquipment + getAllValueSR(): List<AbstractSR> + getAllValueLocation(): List<Location> + getAllValueEmployee(): List<Employee> + getAllValueMedicalEquipment(): List<MedicalEquipment> + createTableSR(): void + createTableLocation(): void + createTableEmployee(): void + createTableMedicalEquipment(): void + dropTableSR(): void + dropTableLocation(): void + dropTableEmployee(): void + dropTableMedicalEquipment(): void + dropAll(): void + restoreTableSR(): void + restoreTableLocation(): void + restoreTableEmployee(): void + restoreTableMedicalEquipment(): void + restoreAll(): void + isClient(): bool

+ isRemote(): bool + isEmbedded(): bool





- Used JDBC API for low-level access of databases
- Data access object (DAO) pattern using Generics
 - Data access object interface (e.g. IDatabase)
 - Data access object concrete class (e.g. MedicalEquipmentDaoI, MedicalEquipmentMongo)
 - o Model object (e.g.
 MedicalEquipment)





+ addValue(MedicalEquipment): void + deleteValue(MedicalEquipment): void + updateValue(MedicalEquipment): void

MedicalEquipmentMongo

- + getValue(): MedicalEquipment + getAllValue(): List<MedicalEquipment>
- + createTable(): void
- + dropTable(): void
- + restoreTable(): void

- Classes to manage database connections
- MongoDB class
 - For cloud-based MongoDB database
- **ConnectionManager** class
 - For embedded and client-server Apache Derby databases
 - Switches between both connections

ConnectionManager

- conn: ConnectionManager
- embeddedConnection: Connection
- clientConnection: Connection
- useClient: boolean

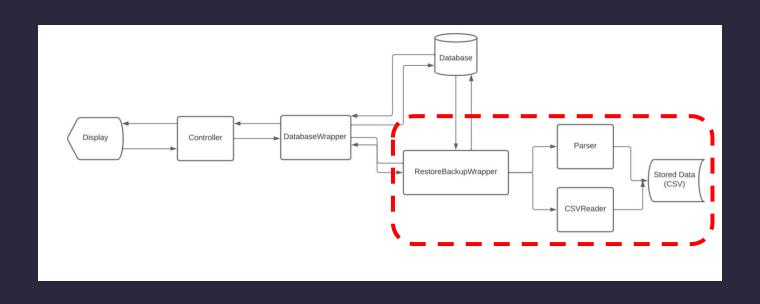
getInstance(): ConnectionManager getConnection(): Connection setConnectionStrategy(client) closeConnection()

MongoDB

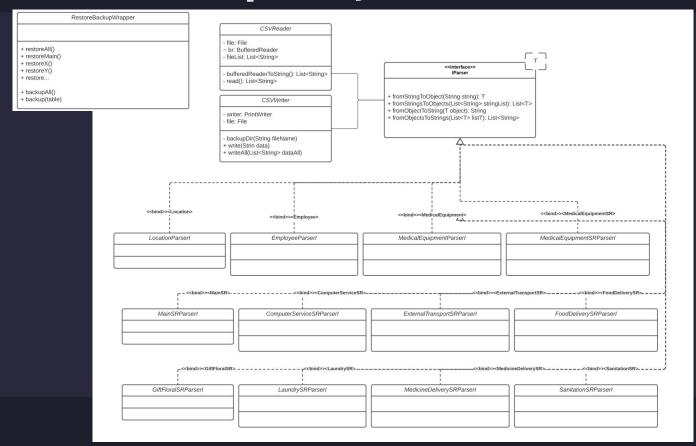
- + MongoClient: MongoClient
- + DB: BDBMongo
- DBCollection LocationMongo

getBDBMongo(): DB getConnection(): Connection convert(Location) : DBObject

Restore/Backup Subsystem

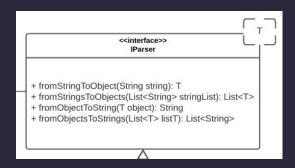


Restore/Backup Subsystem

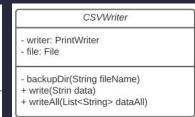


Restore/Backup Subsystem

- Purpose of subsystem was to read from (restore) and write to (backup) CSV files.
- **RestoreBackupWrapper** class
 - Interface between rest of back-end system and the subsystem (Facade pattern)
- **IParser** generic interface class
 - Converted from List of Strings to Collection of Objects
 - Converted from Collection of Object to List of Strings
- Implemented low-level CSVReader and CSVWriter classes



CSVReader - file: File - br: BufferedReader - fileList: List<String> - bufferedReaderToString(): List<String> - read(): List<String>



Demo