Smart Employee Management System (SEMS)

TEAM PENTAD X-5

Overview

Motivation for enhancement

Architecture Enhancement

Enhanced Sub-system

Enhanced Use cases

Enhanced Diagrams

Plans for Testing

Mitigation Tactics

Alternatives for Enhancement

Potential Risks

Concurrency

Testing for impacts of Interaction

Lessons Learned

Motivation for enhancement

Automates

The detection of failures

The dealing of failures

Minimizing downtime within the system

Creates a more attractive product for end users

More Reliable

More Available

Architecture Enhancement

Layered Architecture

User Actions from Client Tier

Logger feature in Business Layer

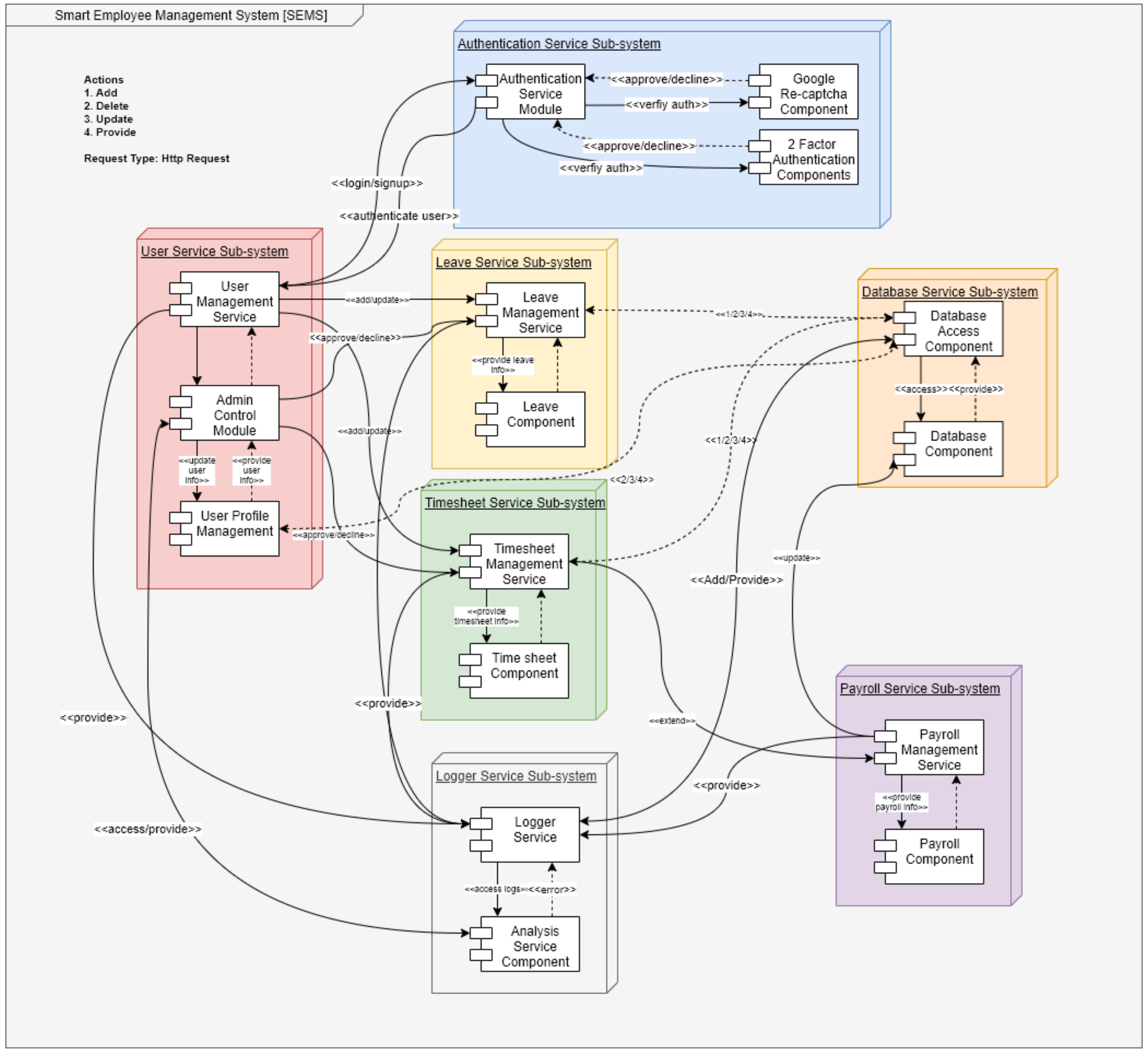
Logs stored in Data Tier

MVC (Model View Controller)

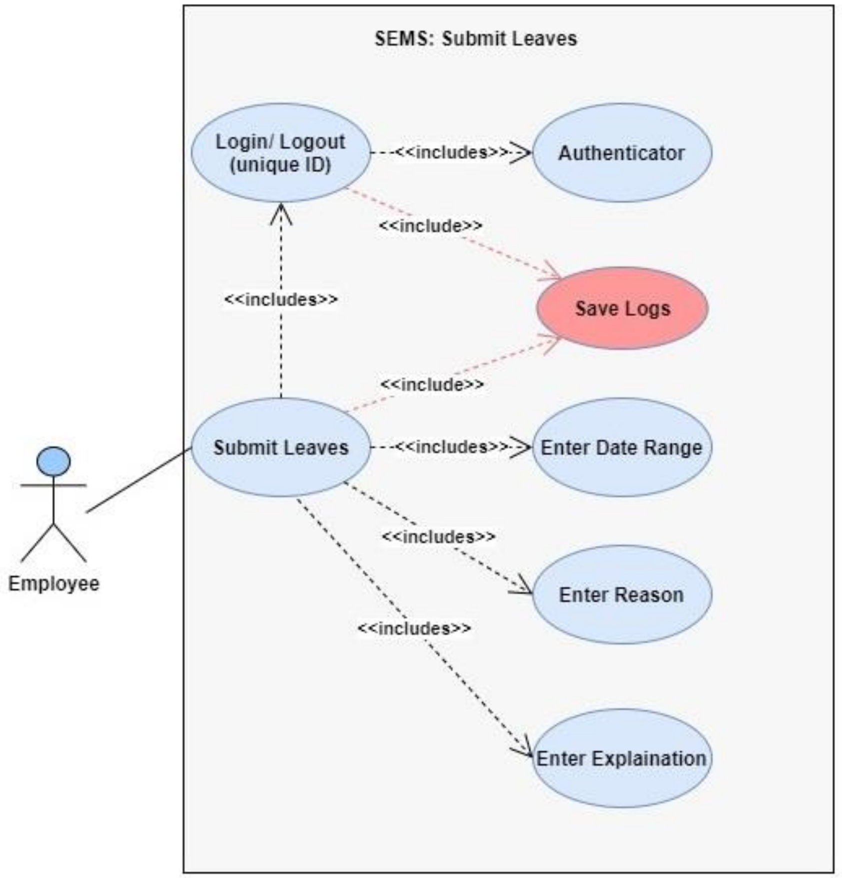
Extra Logger View

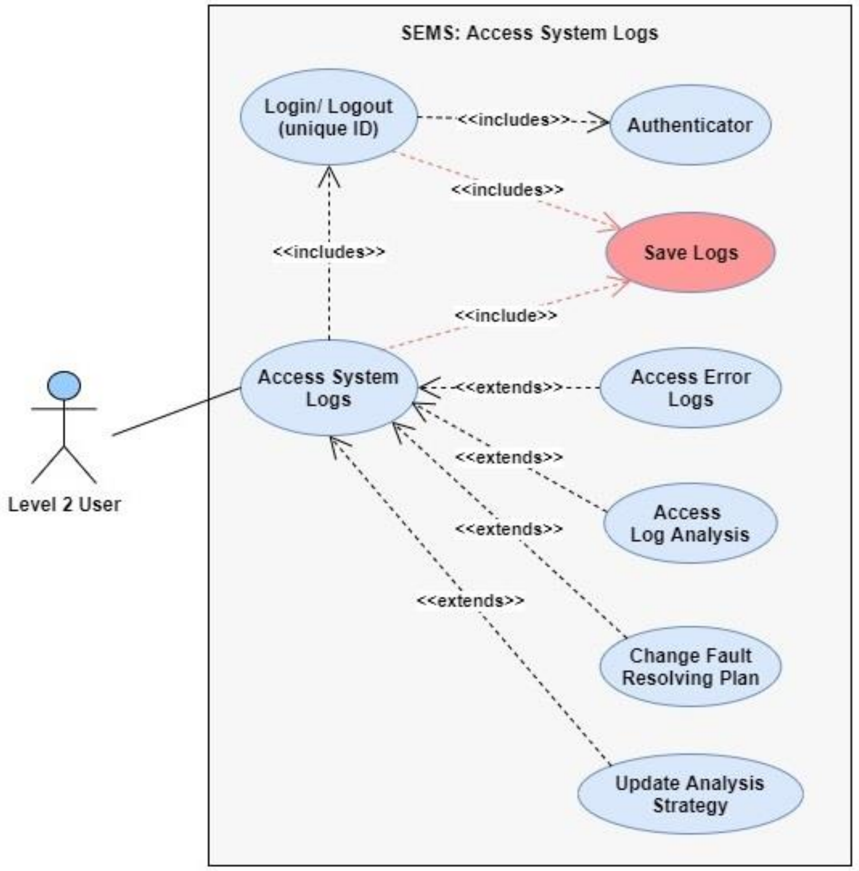
Additional Controller Feature

Enhanced Sub-system Diagram

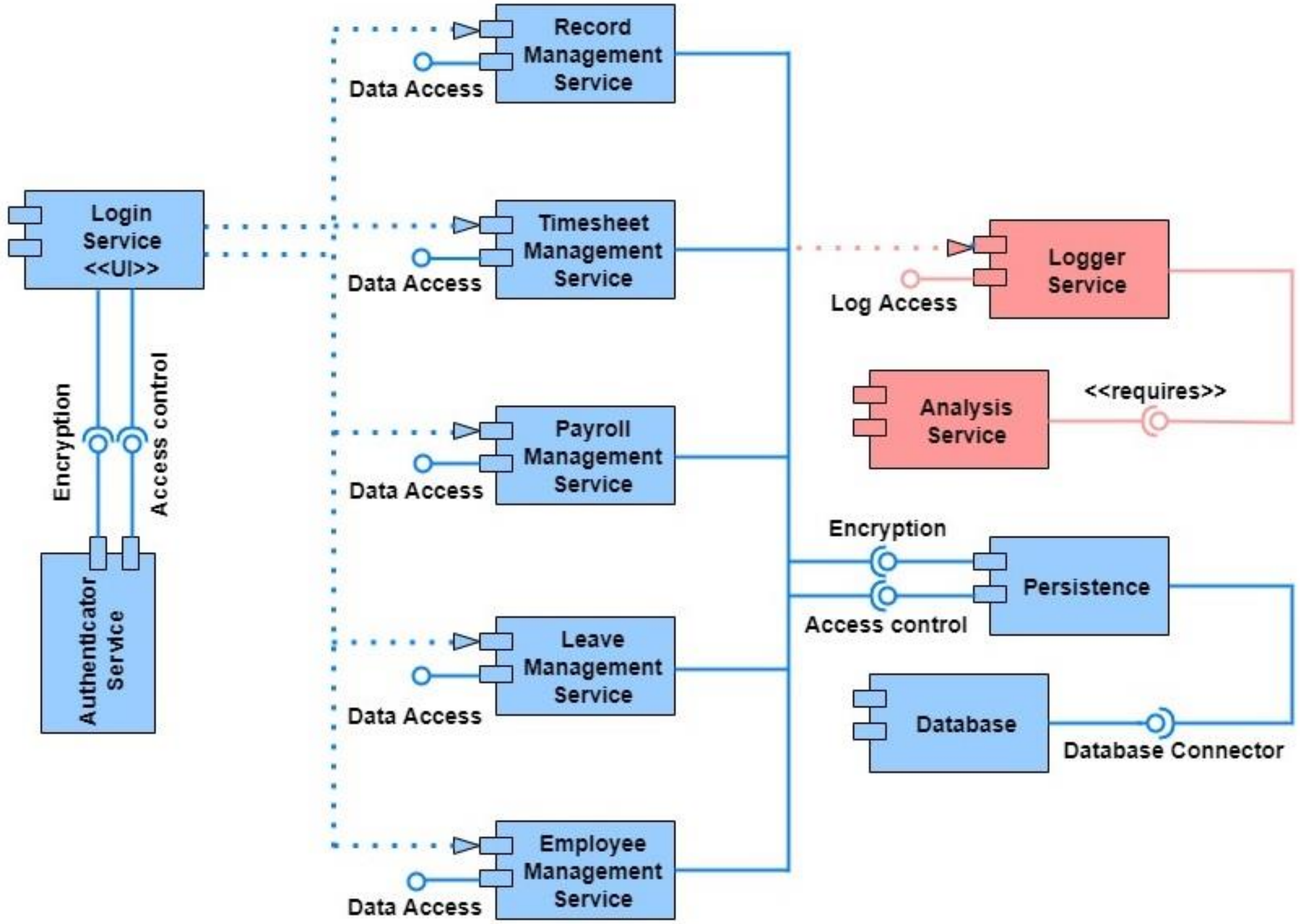


Enhanced Use cases:

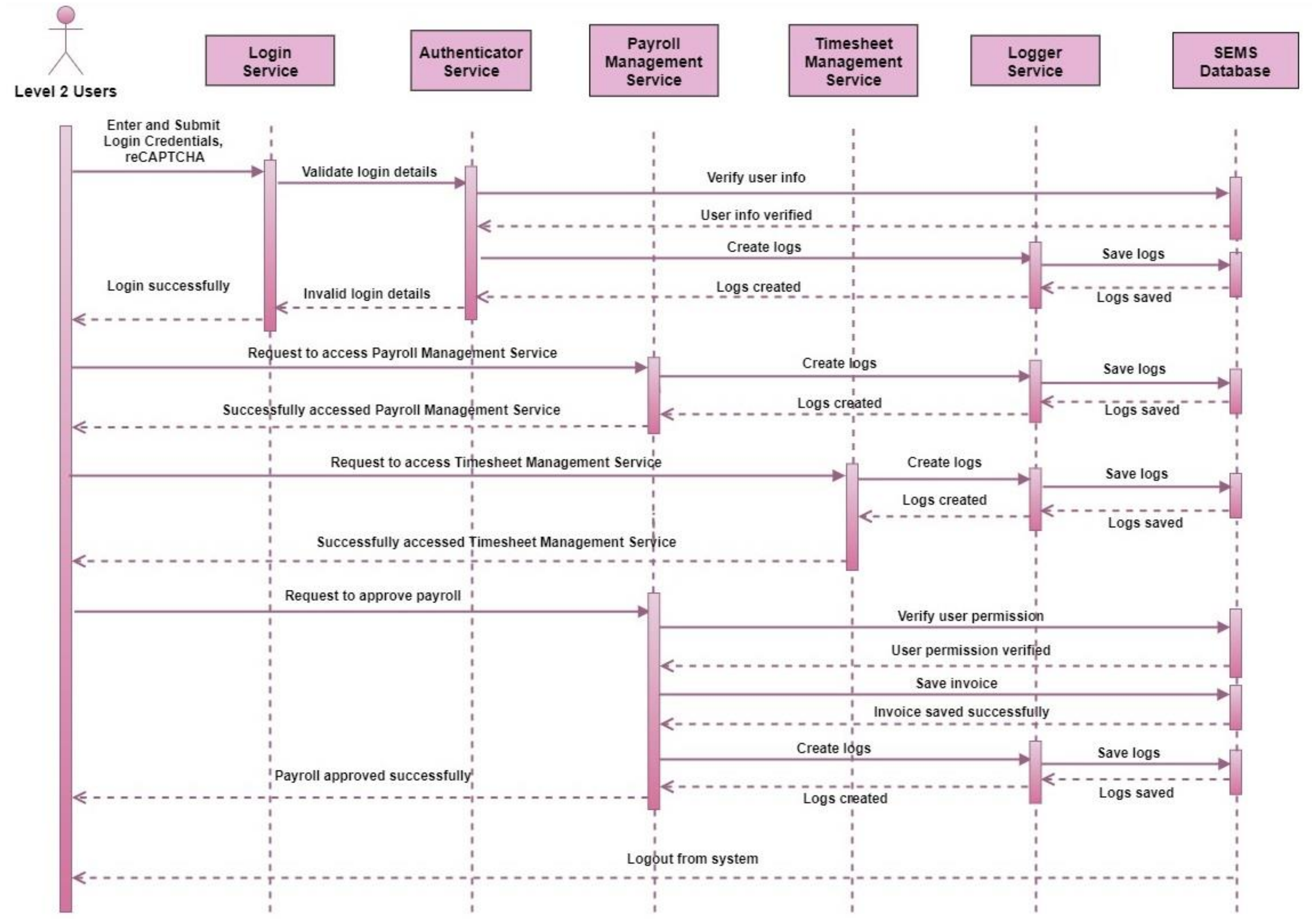




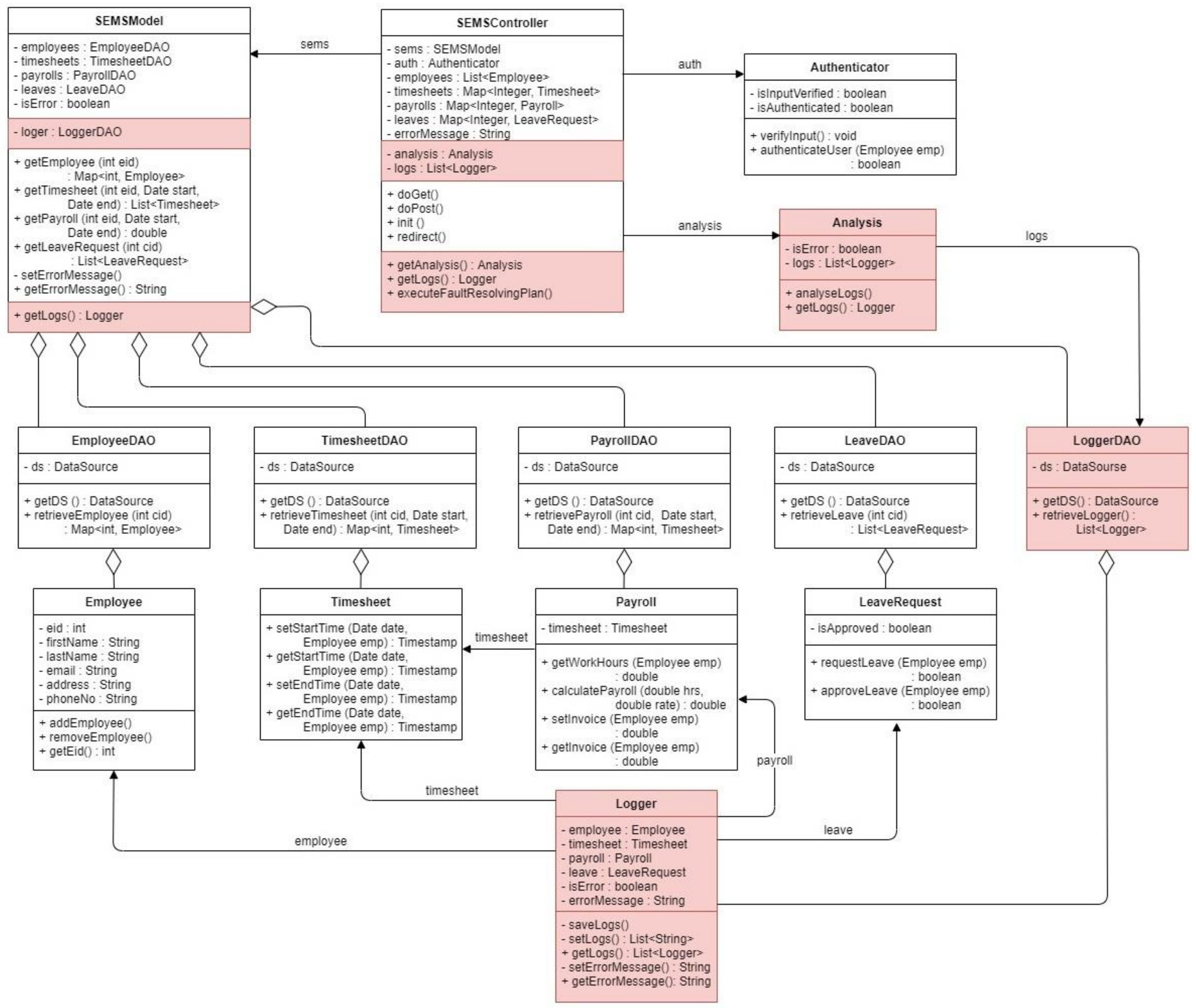
Added Components for the Enhancement



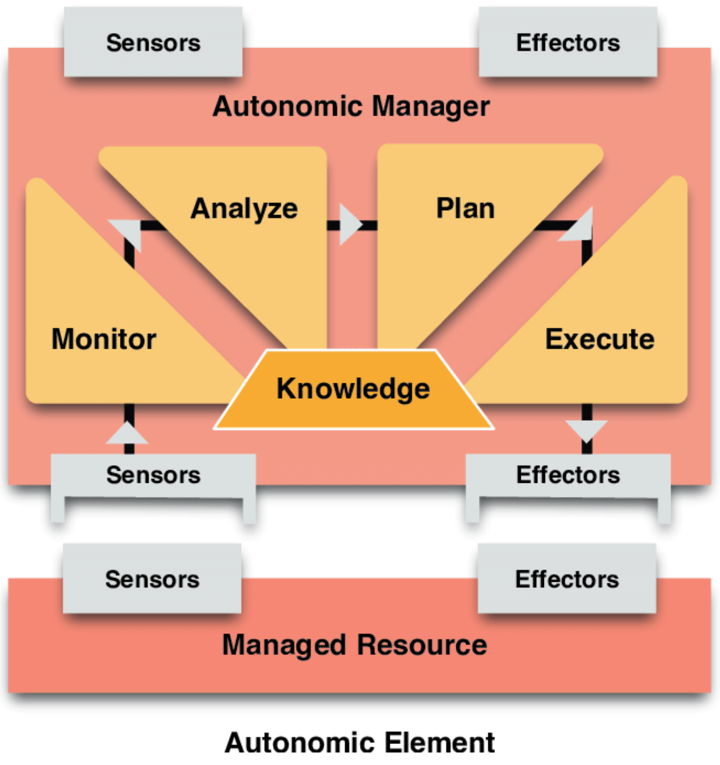
Enhanced Sequence Diagram Representing Control of Data-Flow



Enhancement of the Class Diagram


SEMSModel
- employees : EmployeeDAO
- timesheets : TimesheetDAO
- payrolls : PayrollDAO
- leaves : LeaveDAO
- isError : boolean
- loger : LoggerDAO
+ getEmployee (int eid) : Map<int, Employee>
+ getTimesheet (int eid, Date start, Date end) : List<Timesheet>
+ getPayroll (int eid, Date start, Date end) : double
+ getLeaveRequest (int cid) : List<LeaveRequest>
- setErrorMessage()
+ getErrorMessage() : String
+ getLogs() : Logger
sems
SEMSController
- sems : SEMSModel
- auth : Authenticator
- employees : List<Employee>
- timesheets : Map<Integer, Timesheet>
- payrolls : Map<Integer, Payroll>
- leaves : Map<Integer, LeaveRequest>
- errorMessage : String
- analysis : Analysis
- logs : List<Logger>
+ doGet()
+ doPost()
+ init ()
+ redirect()
+ getAnalysis() : Analysis
+ getLogs() : Logger
+ executeFaultResolvingPlan()
auth
Authenticator
- isInputVerified : boolean
- isAuthenticated : boolean
+ verifyInput() : void
+ authenticateUser (Employee emp) : boolean
analysis
Analysis
- isError : boolean
- logs : List<Logger>
+ analyseLogs()
+ getLogs() : Logger
logs
EmployeeDAO
- ds : DataSource
+ getDS () : DataSource
+ retrieveEmployee (int cid) : Map<int, Employee>
TimesheetDAO
- ds : DataSource
+ getDS () : DataSource
+ retrieveTimesheet (int cid, Date start, Date end) : Map<int, Timesheet>
PayrollDAO
- ds : DataSource
+ getDS () : DataSource
+ retrievePayroll (int cid, Date start, Date end) : Map<int, Timesheet>
LeaveDAO
- ds : DataSource
+ getDS () : DataSource
+ retrieveLeave (int cid) : List<LeaveRequest>
LoggerDAO
- ds : DataSourse
+ getDS() : DataSource
+ retrieveLogger() : List<Logger>
Employee
- eid : int
- firstName : String
- lastName : String
- email : String
- address : String
- phoneNo : String
+ addEmployee()
+ removeEmployee()
+ getEid() : int
Timesheet
+ setStartTime (Date date, Employee emp) : Timestamp
+ getStartTime (Date date, Employee emp) : Timestamp
+ setEndTime (Date date, Employee emp) : Timestamp
+ getEndTime (Date date, Employee emp) : Timestamp
timesheet
Payroll
- timesheet : Timesheet
+ getWorkHours (Employee emp) : double
+ calculatePayroll (double hrs, double rate) : double
+ setInvoice (Employee emp) : double
+ getInvoice (Employee emp) : double
LeaveRequest
- isApproved : boolean
+ requestLeave (Employee emp) : boolean
+ approveLeave (Employee emp) : boolean
payroll
timesheet
leave
employee
Logger
- employee : Employee
- timesheet : Timesheet
- payroll : Payroll
- leave : LeaveRequest
- isError : boolean
- errorMessage : String
- saveLogs()
- setLogs() : List<String>
+ getLogs() : List<Logger>
- setErrorMessage() : String
+ getErrorMessage(): String

Mitigation Tactics:



Monitor
- User logs

Analyze
- Error messages

Plan
- Retry Failed Operations

Execute
- Redirect

Alternatives for Enhancement

Smarter Logging Feature

Temporary Logger

Potential Risks

Maintainability:

Ease of Access for users, increased maintenance for developers.

Notifying component changes across the system.

Evolvability:

New Component needs to access the sub-system.

Testability:

Easily test and debug current and future functionality.

Performance

No major performance draw back.

Increased network traffic.

Increase database read/write. Increased access times.

Security & Privacy:

No Security issues. No data leak through the new component.

Enhancement used to determine system failure and not to monitor data transfer.

Concurrency in SEMS

User Concurrency	• Serving multiple users at the same time
Component Concurrency	• A component can be used by many users simultaneously
Example:	• Multiple users should be able to login to the system simultaneously and access the different components of the system while others are also using them

Testing impacts of interactions

Self-healing loop monitors components and executes solutions

Interactions must be correct as the function of the self-management loop depends on the data it takes as input

Testing to ensure correct interactions

Check if logs are correct

Check if faults are detected

Check if solutions are deployed correctly to the components

Lessons Learned

How to implement enhancements into an existing system

The impacts of such enhancements

The different components of a self-healing loop

What data is monitored

How is that data analyzed

How complex dealing with faults can be