LOW-LEVEL DESIGN REPORT

CMPE 492

Senior Design Project II

Çağla Yıldız 1431404597

Deniz Zeynep Ersoy 2743946522

Tuğçe Nilay Öztekin 1004013348

Table of Contents

1. INTRODUCTION

In today’s competitive environment, businesses tend to overlook the motivation of employees to produce products. However, businesses may increase employee motivation and overall productivity by implementing workplace gamification strategies. Today, gamification performance management has become a widely used term.

According to the research conducted by TalentLMS, 89% of employees state that gamification increases their sense of productivity at work. Furthermore, in another research, if a business utilizes gamified activities, 69% of its employees are more likely to remain longer than three years. These figures demonstrate the effectiveness of gamification in retaining company workers. In this project, we propose the development of an application that aims to increase productivity in production by increasing positive competition through gamification.

Lead the Board exists to create a work environment where employees can engage with positive competition, providing an increase in efficiency, productivity and fun.

Lead the Board is a software that integrates gamification in manufacturing to offer rewards to employees for their contribution to their work. It gives managers and employees the ability to track their progress and reward employees who are bringing their full potential to the game. With 3 different user types such as admin, manager, and employee, different modules will be provided with position-unique content. Employees will earn points and badges based on metrics such as production, operation type, and difficulty level and be placed on a leaderboard. Employees who reach specified rankings will be rewarded.

The completion of the tasks by the operators will be tracked and counted through sensors. As a result, the operator will earn as many points as the task he/she performs. Also, a feed page will be constructed for everyone to share their achievements, thoughts, ideas, and comments.

1.1 Object Design Trade-Offs

In this section, the object design trade-offs will be discussed, which compare reliability, compatibility, performance, maintainability, functionality and usability.

1.1.1. Reliability vs. Compatibility

One of the key trade-offs in object design is between reliability and compatibility. A flexible design may be easier to implement and maintain as a compatible way but may not be reliable enough. On the other hand, a more reliable design may be more complex and harder to implement in a compatible way but may be better suited for Lead the Board's requirements. Since there are many important and sensitive data need to be kept track of such as points, workload, rank, badge etc., we prioritize reliability over compatibility.

1.1.2. Performance vs Maintainability

Between performance and maintainability, there is another significant trade-off. A design that puts performance first might be quicker and more effective, but it might also be more challenging to maintain and modify. Due to the high amount of users of Lead the Board, performance is a key objective even if maintainability is a major concern.

1.1.3 Functionality vs. Usability

When we compare functionality and usability, usability is a more important concern of Lead the Board. Since our users come from different backgrounds and different education levels, the interface and design of Lead the Board in both mobile and desktop should be straightforward enough for all types of users to interact with the system.

1.2 Interface documentation guidelines

The purpose of this section is to provide guidelines for documenting the interfaces between the different components and modules of the system. These guidelines aim to ensure that the interfaces are well-defined, clearly documented, and consistent throughout the system.

1.2.1 Interface Documentation Format

All classes, attributes, and methods in this report are designated using camel case. Class titles begin with a capital letter, while others do not. The structure for class interface descriptions is as follows:

class ClassName
The description of the class.
Attributes
typeOfAttribute nameOfAttribute ...
Methods
returnType methodName(parameters): Method explanation if it is needed ...

1.3 Engineering Standards

In Lead the Board, UML diagrams are utilized to model the software systems and IEEE guidelines followed when documenting the project designs. The UML is a crucial component of the process of designing object-oriented software, so it is a commonly used way to generate necessary diagrams. By adhering to these standards, we ensure that the designs are of high quality and easily understood by all members of the project team.

1.4 Definitions, Acronyms, and Abbreviations

It is crucial to clarify any technical terms, acronyms, or abbreviations used in the report in order to prevent any misunderstanding or ambiguity. The definitions used throughout the report should be precise, succinct, and uniform.

There may be use of the following meanings, acronyms, and abbreviations in this document:

- Object Design: Converting a software system's high-level design into a detailed low-level design through the identification of new problem objects and the improvement of old ones.
- UML: Software engineering uses "The Unified Modeling Language," a standardized modeling language, to depict software systems visually.
- IEEE: The professional association known as the Institute of Electrical and Electronics Engineers, which creates and disseminates guidelines for a range of industries, including software engineering.
- API: A collection of tools and protocols called "Application Programming Interface" are used to create software applications.
- GUI: Graphical User Interface, a kind of user interface that enables users to communicate with software systems using graphical components like buttons and menus.
- OOP: Object-Oriented Programming, a paradigm for programming software that arranges systems around things with traits and behaviours.
- CRUD: Create, Read, Update, Delete, a group of fundamental operations frequently used in database systems.
- MVC: Model View Controller architecture.
- UI: User Interface

2. PACKAGES

Lead the Board is divided into two parts at the highest level: Client and Server.

2.1 Server

2.1.1 Model

- User
- Manager
- Operator

- Admin
- Badge
- Skill
- Department
- Operation
- Product
- Machine
- Leader Board
- RegistrationRequest
- Point

2.1.2 Controller

- UserController
- ManagerController
- OperatorController
- AdminController
- BadgeController
- PointController
- RegistrationController
- SkillController
- DepartmentController
- OperationsController
- ProductController
- MachineController
- LeaderBoardController

2.1.3 Service

- ManagerService
- OperatorService

- BadgeService
- PointService
- RegistrationService
- SkillService
- DepartmentService
- OperationService
- ProductService
- MachineService
- LeaderBoardService

2.1.4 Repository

- UserRepository
- ManagerRepository
- OperatorRepository
- AdminRepository
- BadgeRepository
- PointRepository
- SkillRepository
- DepartmentRepository
- OperationsRepository
- ProductRepository
- MachineRepository

2.2 Client

2.2.1 Operator Components

- LoginComponent
- SignComponent
- BadgeComponent

- BadgeDetailComponent
- LeaderboardComponent
- QRComponent
- TaskSelectionComponent
- TaskInProgressComponent
- UserProfileComponent

2.2.2 Admin Components

- OverviewComponent
- OperationsComponent
- MachinesComponent
- DepartmentsComponents
- SkillsComponent
- SkillAssignmentComponent
- TaskAssignmentComponent
- ProductCatalogComponent
- BadgeOperationComponent
- RegistrationRequestsComponent
- ManagerRegistrationComponent
- OperatorRegistrationComponent
- UsersComponent

2.2.3 Manager Components

- OverviewComponent
- OperationsComponent
- TaskAssignmentComponent
- OperatorsComponent
- LeaderBoardComponent
- ProductCatalogComponent

3. Class Interfaces

3.1 Server

3.1.1 Model

class User
This class is used to represent and store information about user.
Attributes
private String name private String surname private long ID private String password private String department
Methods
Getter and setter methods.

class Manager
This class is used to represent and store information about manager. This class inherits User class.
Attributes
private Operator[] operatorsInCharge private Product[] productsInCharge
Methods
Getter and setter methods.

class Operator
This class is used to represent and store information about operator. This class inherits User class.
Attributes
private int totalPoints private Badge[] badges private Dictionary<Skill skill, int level> skills Private Operation[] operations
Methods
Getter and setter methods.

Class Admin
This class is used to represent and store information about admin. This class inherits User class.
Attributes
Private string id Private string name
Methods
Getter and setter methods.

class Badge
This class is used to represent and store information about badge.
Attributes
private String name private long ID private LocalDate duration private String description private LocalDate startTime private LocalDate endTime
Methods
Getter and setter methods.

class Point
This class is used to represent and store information about point.
Attributes
private int point private Operator operator
Methods
Getter and setter methods.

class Skill
This class is used to represent and store information about skill.
Attributes
private int id private String name private int level private string description
Methods
Getter and setter methods.

class Department
This class is used to represent and store information about department.
Attributes
private int id private String name private Manager manager private Operator[] operators private Product[] products
Methods
Getter and setter methods.

class Machine
This class is used to represent and store information about machines.
Attributes
private int id private String name private URL qrCode private String descriptions
Methods
Getter and setter methods.

class Product
This class is used to represent and store information about products.
Attributes
private int id private String name private String type
Methods
Getter and setter methods.

class LeaderBoard
This class is used to represent and store information about leader board.
Attributes
private Operator operator private int totalPoint
Methods
Getter and setter methods.

Class Operations
This class is used to represent and store information about operations.
Attributes
private int id private String name private int difficultyLevel private int points private Employee assignedTo private Machine machine private Dictionary skills Private boolean isCompleted
Methods
Getter and setter methods.

class RegistrationRequest
This class is used to represent and store information about registration requests that admin accept or declines.
Attributes
private Operator operator private boolean approvalStatus
Methods
Getter and setter methods.

3.1.2 Service

class ProductService
Controls the logic of product operations.
Attributes
Methods
Product addProduct(Product product): Adds new product to the boolean deleteProduct(long id): Deletes the product with given id. If the operation is successful, the method return 0. If it is failed, it returns 1. Product updateProduct(long id, Product product) Product[] addProductToManager(Product product, Manager manager): Adds an already existing product to given manager. Product[] getProductList() Product getProductById(long id) Product[] getProductListByDepartmentId(long id)

class OperatorService
Controls the logic of operator operations.
Attributes
Methods
Operator addOperator(Operator operatorToBeAdded): Adds a new operator. Boolean deleteOperator(long id): Deletes the operator with given id. If the operation is successful, the method return 0. If it is failed, it returns 1. Operator updateOperator(long id): Updates the operator with given id. Badges[] getBadges(long id): Returns the earned and in-progress badges of specific operator whose id is given. Operations[] getAssignedOperation(long id): Returns the assigned operations

class DepartmentService
Controls the logic of department operations.
Attributes
Methods
Department addDepartment(Department deptToBeAdded): Adds new department that given as parameter This method also matches product and department. Boolean deleteDepartment(long id): Deletes the department with given id. Department findDepartment(long id): Returns the department with given id.

class SkillService
Controls the logic of skill operations.
Attributes
Methods
Skill addSkill(Skill skill): Adds a new skill. Boolean deleteSkill(long id) Skill updateSkill(long id, Skill skillToBeUpdated) int assignSkill(Operator operator, Skill[] skills): Assigns given skill(s) to given operator. If the operation is successful, the method return 0. If it is failed, it returns 1. Skill[] getSkillList() Skill[] getRequiredSkillsOfTheOperationById(long id): Operations have required skills that operator must meet to assign to the operation.

class OperationService
Controls the logic of operation type’s operations.
Attributes
Methods
int assignOperation(Operator operator, Operation[] operation): Assigns given operation(s) to given operator.If the operation is successful, the method return 0. If it is failed, it returns 1,

class ManagerService
Controls the logic of manager operations.
Attributes
Private Product productRepo
Methods
Manager addManager(Manager manager) Operator[] addOperator (Operator operator): Adds new operator to the operator list of the manager. Product[] addProductToManager(Product product, Manager manger): Adds an already existing product to given manager.

class BadgeService
Controls the logic of badge operations.
Attributes
Methods
Badge addBadge(string badgeName, string description, int minPointRange, int maxPointRange, string validityTİmeType, IMAGE, Date beginDate, Date endDate): Creates a new badge with given parameters. List<Badge> getBagdeList(): Returns all badges. Badge getBadgeInformationById(long Id): Returns a specific badge’s information. Boolean deleteBadge(long id): Deletes badge with given ID. Badge updateBadge(long id, params[]): Update badge with given ID and new given parameters. Badge[] getBadgesByOperatorId(long id)

class MachineService
Controls the logic of machiene operations.
Attributes
Methods
Machine add Machine (Machine newMachine): Creates a new Machine with given parameters. List<Machine> getMachineList(): Returns all machines. Machine getMachineInformationById(long Id): Returns a specific machine’s information. Boolean deleteMachiene(long id): Deletes machine with given ID. Machine updateMachine(long id, params[]): Update machine with given ID and new given parameters.

class LeaderBoardService
Controls the logic of Leaderboard operations.
Attributes
Methods
LeaderBoard addLeaderBoard (LeaderBoard newLeaderBoard): Creates a new LeaderBoard with given parameters. LeaderBoard getLeaderBoardInformationById(long Id): Returns a specific LeaderBoard’s information. LeaderBoard updateLeaderBoard (long id, params[]): Update LeaderBoard with given ID and new given parameters. LeaderBoard getLeaderBoardInGivenTimePeriod(LocalDate start, LocalDate end) Operator[] getOperatorsInLeaderboard(int id)

class PointService
Controls the logic of point operations.
Attributes
Methods
int calculatePoints(Operator operator): Calculates the operators’s -whose id is given- completed operations’ total point. int returnPoints(Date beginDate, Date endDate, int id): Calculates the operator’s -whose id is given- total points according to completed tasks between given time interval.

class RegistrationService
Controls the logic of Registration operations.
Attributes
Methods
Registration createRegistration (Registration newRegistration): Creates a new Registration with given parameters. Registration getRegistration InformationById(long Id): Returns a specific Registration information. Registration[] getAcceptedRegistrations() Registration[] getDeclinedRegistrations() Operator acceptOperatorRegistration(Registration acceptedRegistration)

3.2 Client

3.2.1 Operator Components

class LoginComponent
This class shows the login screen and enable user to login to the system
Attributes
Methods
Boolean login(string email, string password): Logs in the user with given e-mail and password. If the credentials are correct, it returns true, otherwise false.

class SignUpComponent
This class shows the sign-up screen and enable user to sign up to the system
Attributes
Methods
Boolean signUp(string name, string surname, string department, string email, string password): Signs up the user with given information. If the operation is successful, it returns true, otherwise false.

class BadgeComponent
This class represents the badge screen of the operator. It contains all available badges. Earned badges and badges in progress are shown differently.
Attributes
Badge[] badgeList Badge[] earnedBadges Badge[] badgesInProgress
Methods
Badge[] getAllAvailableBadges() Badge[] getEarnedBadgesOfOperator(long operatorId) Badge[] getBadgesInProgressofOperator(long OperatorId) void shareBadge(long badgeId)

class BadgeDetailComponent
This class shows the badge detail screen
Attributes
Methods
Badge returnBadgeInfo(long id): Returns a badge info whose id is given. Badge updateBadge(long id, params[]): Update a Badge with given ID and new given parameters.

class LeaderBoardComponent
This class represents the Leaderboard component of the Lead the Board.
Attributes
Operator[] operatorsInTheLeaderboard Operator[] top10Operators
Methods
Operator[] getOperatorsInTheLeaderboard() Operators[] getTop10Operators() Operators[] getLeaderboardOnTheGivenTimePeriod(Date start, Date end)

class OperationSelectionComponent
This class shows the operation selection screen.
Attributes
Operation selectedOperation Operation[] operationOptions
Methods
Operation selectOperation(long id) Operation[] getOperationOptionsOfOperator(long operatorId)

class OperationIsProgressComponent
This class shows the progress screen of operation.
Attributes
Operation selectedOperation int gainedPoints Badge[] gainedBadges int totalTime boolean isOperationCompleted DateTime startTime DateTime endTime
Methods
getOperationDetailedInformation(long operationId) Point endTask(long operationId) int calculateTotalPoints()

class UserProfileComponent
This class represents the profile component of the operator.
Attributes
int leaderboardRankOfOperator long operatorId
Methods
Operator getOperatorInformation(long id) int getLeaderboardRank(long id) int getDailyProgressPoint(long id) int getTotalPointsOfOperator(long id) int getWeeklyPointsOfOperator(long id)

3.2.2 Manager Components

class OverviewComponent
This class represents the general information about manager’s own department.
Attributes
Product [] products Operator[] operators Operation[] operations User[] numberOfUsersOnline int totalProductsThatMade int totalPoint Calender calendarComponent
Methods
Product[] getProductsOfManager() Operator[] getOperatorListOfManager() Operation[] getOperationListOfManager() User[] getOnlineUsers() int getTotalProductNumber() int getTotalPoints() Date getCurrentDate()

class OperationsComponent
This class represents details and requirements of an operation.
Attributes
Operation[] operations Product[] products Machine[] machines Skill[] skills int point int level
Methods
Operation createOperation(Product product, Operation operation, Machine machine ,Skill[] skills, int point, int level) Operation selectOperation(Operation operation) Operator selectOperator(Operator operator) boolean deleteOperation(Operation operation) Operation updateOperation(Operation operation)

class TaskAssignmentComponent
This class represents assignment of an operation to an operator
Attributes
Operation[] operation Operator[] operator Operation[] assignedtasks Operation selectedOperation Operator selectedOperator
Methods
assignTaskToOperator(Operation taskId, long operatorId)

class OperatorsComponent
This class shows the general information of all employees working in the department.
Attributes
Operator[] operators Operator selectedOperator
Methods
Operator getAllOperators() Operator getOperatorsOfManager(long managerId) Operator sortOperatorsLists(Operator[] operatorList) Operator getOperatorById(long operatorId) Operator searchOperator(searchParams[])

class LeaderBoardComponent
This component displays the leaderboard of the operators based on their performance.
Attributes
Operator[] currentLeaderBoard Date selectedDate Date startDate Date endDate
Methods
void displayCurrrentRanking() LeaderBoard getRankingAtDate(Date date) LeaderBoard getRankingAtTimePeriod(Date start, Date end)

class ProductCatalogComponent
This class shows the product catalogue.
Attributes
Product[] products
Methods
Product getProductList() Product addProduct(Product newProduct) Product updateProduct(long productId, updatedProduct) Product deleteProduct(long productId)

3.2.3 Admin Components

class OverviewComponent
This class represents the general information about manager’s own department.
Attributes
Product [] products Operator[] operators Operation[] operations User[] numberOfUsersOnline int totalProductsThatMade int totalPoint Calender calendarComponent
Methods
Product[] getProductsOfManager() Operator[] getOperatorListOfManager() Operation[] getOperationListOfManager() User[] getOnlineUsers() int getTotalProductNumber() int getTotalPoints() Date getCurrentDate()

class OperationsComponent
This class represents details and requirements of an operation.
Attributes
Operation[] operations Product[] products Machine[] machines Skill[] skills int point int level
Methods
Operation createOperation(Product product, Operation operation, Machine machine ,Skill[] skills, int point, int level) Operation selectOperation(Operation operation) Operator selectOperator(Operator operator) boolean deleteOperation(Operation operation) Operation updateOperation(Operation operation)

class TaskAssignmentComponent
This class represents assignment of an operation to an operator
Attributes
Operation[] operation Operator[] operator Operation[] assignedtasks Operation selectedOperation Operator selectedOperator
Methods
assignTaskToOperator(Operation taskId, long operatorId)

class UsersComponent
This components displays information of all users.
Attributes
List<User> allUsers
Methods
User getUserById(long id) User updateUserById(long id) List<User> getAllUsers() Void deleteUserById(long id)

class ProductCatalogComponent
This component allows the admin to manage the product catalog used in the system.
Attributes
List<Product> allProducts
Methods
Product findProductById(long id) void deleteProduct(long id) Product editProduct(long id) List<Product> allProduct()

class MachinesComponent
This component displays information about the machines used in the company's operations.
Attributes
Machine [] machines
Methods
Getter and setter methods Machine addMachine() Machine removeMachine() Machine searchMachinesById(int ID) Machine searchMachinesByName(String name)

class DepartmentsComponents
This component displays information about the departments in the compant
Attributes
Department[] departments
Methods
Department addDepartment(Department department) void removeDepartment(Department department) Department editDepartment(long id) List<Department>displayAllDepartments() Department getDepartmentByName(String name) List<Product> getProductList(long id)

class SkillsComponent
This component displays information about the skills required for the company's operations.
Attributes
Skill[] skill
Methods
Skill addSkill(Skill skill) void removeSkill(long id) Skill editSkill(long id) Skill getSkillByName(String name) Skill getSkillByDescription(String description) void remove skill (Skill skill)

class SkillAssignmentComponent
This component allows the admin to assign skills to operators.
Attributes
Operator[] operator Skill[] skill
Methods
void assignSkill(Operator operator, Skill skill) List<Skill> getSkillsByOperator(Operator operator) void removeSkillFromOperator(Skill skill, Operator operator)

class BadgeOperationComponent
This component allows the admin to manage the badges used in the system or create new one.
Attributes
List<Badge> badges
Methods
Badge addBadge() void removeBadge(long id) Badge editBadge(long id)

class RegistrationRequestsComponent
This component shows a list of registration requests from operators and managers.
Attributes
List<RegistrationRequests> registrationReqList
Methods
void approveRequest() void rejectRequest()

class ManagerRegistrationComponent
This component allows the admin to register a new manager in the system.
Attributes
String firstName String lastName String email String password String department
Methods
void register()

class OperatorRegistrationComponent
This component allows the admin to register a new operator in the system.
Attributes
String firstname String lastname String email String password String department
Methods
void register()

3.3 References

"Code of Ethics." Code of Ethics | National Society of Professional Engineers, July 2018, www.nspe.org/resources/ethics/code-ethics.

Jaggavarapu, Manoj. "Presentation Patterns : MVC, MVP, PM, MVVM." Manoj Jaggavarapu WordPress, 2 May 2012, https://manojjaggavarapu.wordpress.com/2012/05/02/presentationpatterns-mvc-mvp-pm-mvvm/.

Fitz-Walter, D. (n.d.). What is gamification? education, Business & Marketing (2021 examples). Retrieved from https://www.gamify.com/what-is-gamification