|  |  |
| --- | --- |
|  | **DOKUZ EYLÜL UNIVERSITY**  **ENGINEERING FACULTY**  **DEPT. OF COMPUTER ENGINEERING** |

BuyArt

# CME 3201 Database Management Systems

# Term Project Report

Phase 1

2019-2020 FALL

2017510079 Berat Özgür Yaman

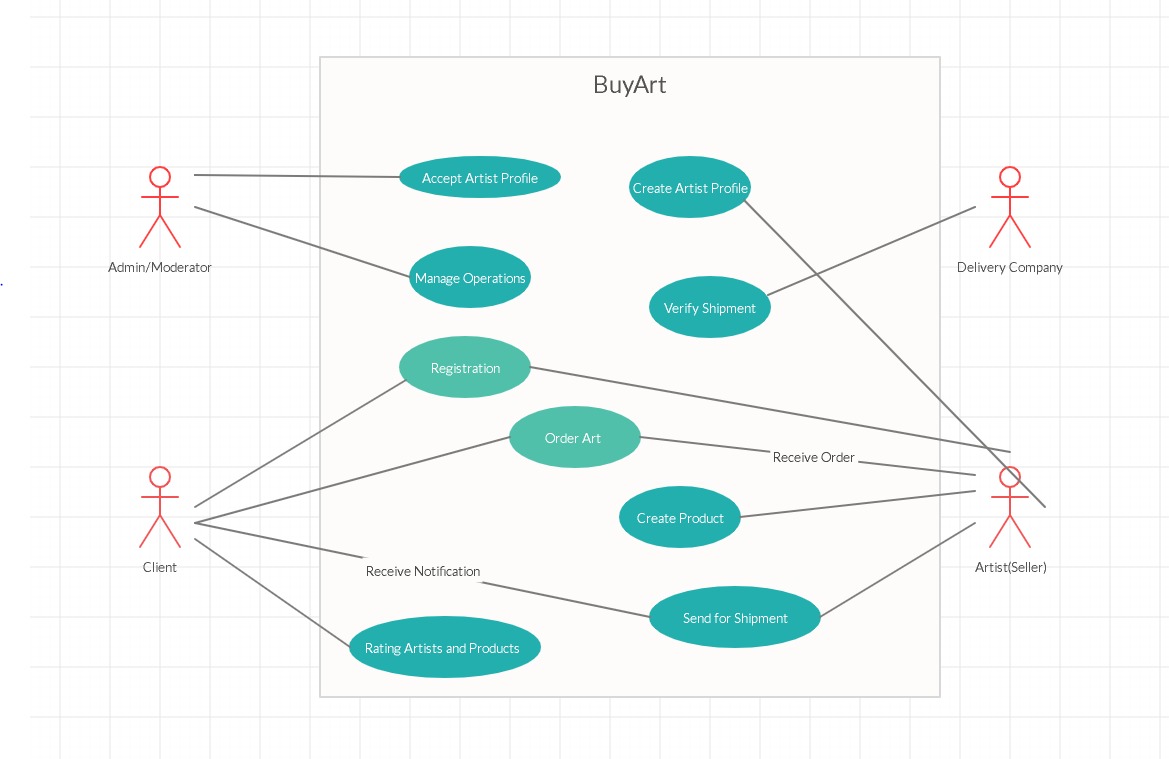
2015510003 Muhammed Ahmet Akkoyun

## Introduction

The name of the project is BuyArt. BuyArt is art marketing system. In Turkey, art marketing is too less than compared to other countries because of there is no known online systems about art marketing. This project’s point is provide this online system for the artists and art lovers. This report contains detailed description of BuyArt’s purpose and technical documentation of software architecture and information about user’s use cases.

## Overview

BuyArt is a website which artists and painters can sell their products and its purpose to take commissions from this transaction operations. Requires user authentication for selling operations. Artist creates a product entry. This product entry has some images for the product and many comments and reviews created by buyers. Buyer can create an order for that product with delivery address. Buyer can create a user account and can save their delivery address and they can see the order history. Generally this project’s point is providing secure art marketing system.



## Assumptions/Constraints/Risks

### Assumptions

There will be two layers, frontend and backend. Frontend will built on Google’s Angular with Material.IO UI package. Backend will built on Node.js with express framework. For DBMS, MySQL will be used holding data. Backend will be RESTful API and Frontend will communicate via REST.

### Constraints

For every type of viewport, ui needs to be look good so, frontend needs to be responsive.

Orders and user information needs to be held secure ways.

When data flow is too much, servers need to be scalable horizontally.

### Risks

Technical issues may occur during the transactions and development process. At this point we will be log everything so we will able to fix it.

It may takes longer time than expected for development because of unpredicted technical or daily life issues.

## Software Architecture

Instructions: Show your details of software architecture. Do you use n-tier architecture? How did you design it in your project? You may draw software architecture diagram (It depicts how a typical software system might interact with its users, external systems, data sources, and services.). How did you provide a connection between database and frontend? Give details about your software architecture.

## Detailed System Design

### Entity-Relationship Diagram

Instructions: Draw the Entity Relationship Diagram applying all rules (relations, attributes, relationships, cardinalities, etc.)

### Relational Algebra Expressions

Instructions: Relational algebra expressions and their SQL for selected five queries.

### Class Diagram

Instructions: Draw your class diagram.

### CRUD Matrix

Instructions: Provide CRUD Matrix (Create, Read, Update, Delete) indicating how the data will be maintained and accessed for all relations like the following table. Did you apply all CRUD operations to all relations?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Relations\Operations** | **Create** | **Read** | **Update** | **Delete** |
| Relation1 | X | X | X | X |
| Relation2 | X | X |  |  |
| Relation3 |  | X |  |  |

**Appendix A: SQL Statements**

Instructions: Provide all SQL statements (create and insert scripts, basic operations on database, view, complex queries, stored procedure, trigger)

**Appendix B: Screenshots**

Instructions: Screen shots of your interfaces with brief declaration for main operations