IBID

Analysis and Design Document

Student: Alexandra Maria Sicobean

**Group: 30431**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 27.03.2023 | <1.0> | Planning and design | Alexandra Maria Sicobean |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

Welcome to the documentation for my bidding application designed specifically for volunteers of the student organization I am currently part of. IBID comes as a solution for decluttering of the office which is way too crowded. This application provides a simple and efficient way for volunteers to bid on their favorite items available in our organization's office.

Gone are the days of the manual bidding process, which can be time-consuming and prone to errors. With our bidding application, volunteers can easily place bids on items they are interested in from the comfort of their own devices.

Our organization recognizes the importance of creating a fair and transparent bidding process for all volunteers, and this application has been designed with those principles in mind. With this application, volunteers can view current bids and track the status of their own bids, ensuring they have a clear understanding of the current bidding situation.

We are confident that this bidding application will enhance the bidding process for our volunteers and ultimately lead to a more organized and efficient system for our organization. Thank you for using our bidding application and we hope this documentation will provide you with all the necessary information to utilize it effectively.

# Elaboration – Iteration 1.1

# Domain Model

As per the project specification for a bidding application for volunteers of a student organization to bid for items in the organization's office, the domain model can be defined as follows:

The system has three main entities: Users, Items, and Bids.

Users can be of two types: Admin and Volunteer.

* Admin has the authority to add new items, update item details, and close the bidding for an item.
* Volunteers can view the available items and bid on them.
* Items have attributes such as Item ID, Item Name, Item Description, Starting Bid Price, Current Bid Price, Bid Start Time, Bid End Time, and Item Image.
* Bids have attributes such as Bid ID, Bid Price, Bid Time, and User ID.
* Users have attributes such as User ID, Username, Email, and Password.

# Architectural Design

## Conceptual Architecture

Building a layered architecture for a .NET web application typically involves separating the application into distinct layers based on their responsibilities. Therefore, in my project I had a presentation, business logic, and data access layer. This approach can help improve code organization, maintainability, and scalability.

1. Here are the steps I followed to build a layered architecture with .NET web application:
2. Define the layers
3. Create projects for each layer
4. Define interfaces
5. Implement the layers
6. Configure Dependency Injection
7. Define Models: Define models for each layer to represent the data they use
8. Implement Controllers in the presentation layer.
9. Implement Services in business layer
10. Implement Repositories in data layer

## Package Design

Diagram

Description automatically generated

## Component and Deployment Diagrams

Diagram

Description automatically generated

# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

Diagram

Description automatically generated

Diagram

Description automatically generated

## Class Design

Diagram

Description automatically generated

# Data Model

*Graphical user interface

Description automatically generated*

# Elaboration – Iteration 2

# Architectural Design Refinement

**Package diagram:**

Diagram

Description automatically generated

# Design Model Refinement

Information Expert: Assign responsibility to the class with the most information needed to fulfill it. Here we have the model classes in the data layer that contain all necessary information and link the application to the database.

Creator: Assign responsibility to the class that has the responsibility to create objects of a given type. Only the controllers in the view layers can create objects of the model classes type and pass the objects to

Controller: Assign responsibility to a class that coordinates activities between other classes. Being a layered architecture, the coordination between classes is done at each level through repositories, services and the controllers in the view layer.

Low Coupling: Assign responsibility to classes that have low coupling with other classes. The low coupling between classes is done through dependency injection.

High Cohesion: Assign responsibility to classes that have high cohesion, meaning that their responsibilities are related and focused.

# Construction and Transition

# System Testing

Test scenario for bidding system integration: This scenario could involve simulating the bidding process for a specific item and verifying that bids are recorded correctly, and that the highest bidder is updated correctly.

Test scenario for payment system integration: This scenario could involve simulating a successful payment for a won item and verifying that the payment is recorded correctly in the system.

Test scenario for inventory system integration: This scenario could involve verifying that the item inventory is updated correctly when an item is won by a user.

Test scenario for user interface and system integration: This scenario could involve testing the interactions between the user interface and the different system components to ensure that users can access the bidding and payment systems as expected.

# Future improvements

1. User feedback system: Implement a system for users to provide feedback on their bidding experience, including ease of use, features they would like to see, and any issues they encountered. This feedback can be used to improve the system for future iterations.

2. Mobile app: Develop a mobile app for the bidding system, allowing users to easily access and bid on items from their smartphones. This can increase user engagement and accessibility.

3. Automatic bidding: Implement an automatic bidding system, which allows users to set a maximum bid for an item and the system will automatically place bids on their behalf up to that maximum amount. This can save time for users and increase the competitiveness of bidding.

4. Auction reminders: Set up a system to send users reminders for upcoming auctions and items they have bid on. This can increase user engagement and ensure they do not miss out on items they are interested in.

5. Improved reporting: Enhance the reporting capabilities of the system to provide more detailed information about auction outcomes, including item popularity, bidding trends, and user behavior. This can help the organization make more informed decisions about future auctions and items to offer.

6. Integration with payment systems: Integrate the system with a payment processing system to allow users to pay for items won directly through the bidding application. This can streamline the process and reduce the need for manual payment processing.

# Bibliography

1. C# fundamentals recap (<https://app.pluralsight.com/library/courses/object-oriented-programming-fundamentals-csharp/table-of-contents>)
2. C# collection fundamentals (<https://app.pluralsight.com/player?course=csharp-collections&author=simon-robinson&name=csharp-collections-fundamentals-m1-welcome&clip=0>)
3. Head first C# book (<https://www.amazon.com/NET-Head-First-Microsoft-Programming/s?ie=UTF8&page=1&rh=n%3A764452%2Cp_lbr_books_series_browse-bin%3AHead%20First>)
4. Web API (<https://app.pluralsight.com/library/courses/implementing-restful-aspdotnet-web-api/table-of-contents>)