

(Website Name)

Name: Mintas Alex Andrei

Group: 30236

Table of Contents

[Deliverable 1 3](#_Toc64843130)

[Project Specification 3](#_Toc64843131)

[Functional Requirements 3](#_Toc64843132)

[Use Case Model 3](#_Toc64843133)

[Use Cases Identification 3](#_Toc64843134)

[UML Use Case Diagrams 3](#_Toc64843135)

[Supplementary Specification 3](#_Toc64843136)

[Non-functional Requirements 3](#_Toc64843137)

[Design Constraints 3](#_Toc64843138)

[Glossary 3](#_Toc64843139)

[Deliverable 2 3](#_Toc64843140)

[Domain Model 3](#_Toc64843141)

[Architectural Design 4](#_Toc64843142)

[Conceptual Architecture 4](#_Toc64843143)

[Package Design 4](#_Toc64843144)

[Component and Deployment Diagram 4](#_Toc64843145)

[Deliverable 3 4](#_Toc64843146)

[Design Model 4](#_Toc64843147)

[Dynamic Behavior 4](#_Toc64843148)

[Class Diagram 4](#_Toc64843149)

[Data Model 4](#_Toc64843150)

[System Testing 4](#_Toc64843151)

[Future Improvements 4](#_Toc64843152)

[Conclusion 4](#_Toc64843153)

[Bibliography 4](#_Toc64843154)

# Deliverable 1

## Project Specification

The (Website Name) website is intended to be used by customers in need of a specific job to be done at their house/apartment by a worker and also by workers in need of a customer for their work. The fields of work that I thought about are: Tile Setter, Painter/Decorator, Locksmith, Electrician, Plumber and more. The (Website Name) primary objective is to facilitate the process of finding a good worker for your desired job and also finding a good customer for each worker so that this interaction is no more fulfilled by scams and displeasures.

[Present the project specification.]

## Functional Requirements

First of all, the user is given the possibility of registration and login because in order to use all the facilities of the website and to perform actions on it you must have a valid account. However, the administrator already has an account and he has a wider control area in which he can operate. There are 3 types of users:

The Customer can register, login, update customer details, find workers, request workers for different jobs, rate the worker after the job is done (or not).

The Worker can register, login, update worker details, find customers, request to do a job for a customer, rate the customers after the job is done (or not).

The Administrator can perform modifications related to the databases, he is a super user.

Before any data is stored into the database, it must be verified to meet the required standards in order to have a easy and clean understanding of it. This verification is done by multiple validators written with great care such that the database cannot be updated until every field of the data has been validated.

## Use Case Model 1

### Use Cases Identification

Use-Case: Login

Level: Subfunction

Primary Actor: User

Main success scenario: Login successful

Use-Case: Register

Level: User Goal

Primary Actor: Customer, Worker

Main success scenario: Register successful

Use-Case: Add/Create User

Level: Subfunction

Primary Actor: User

Main success scenario: User successfully added/created

Use-Case: Update User

Level: Subfunction

Primary Actor: User

Main success scenario: User successfully updated

Use-Case: Delete User

Level: Subfunction

Primary Actor: User

Main success scenario: User successfully deleted

Use-Case: Find Users

Level: Subfunction

Primary Actor: User

Main success scenario: List of users successfully returned

### UML Use Case DiagramsO imagine care conține diagramă, schematic Descriere generată automat

## Supplementary Specification

### Non-functional Requirements

Reliability is an important non-functional requirement that refers to a system’s ability to perform its intended functions consistently, without errors or failures. In my project, I took this into consideration by rigorously testing each component in multiple scenarios. This approach ensured that the system not only performed its intended functions successfully, but also gracefully handled cases where unexpected data or requests were encountered. To achieve this, I created specific exceptions within the application and implemented control flows to handle them effectively. (Reliability)

To enhance usability, my application will feature an intuitive user interface that addresses users with varying levels of technical expertise. Users can interact with the application through intuitive buttons, structured forms, and informative content. The application provides users with suggestive buttons that are labeled clearly, which enables them to generate requests and view relevant information with a single click. Using this approach, the application guarantees an effortless user experience. (Usability)

The application was designed in such a way that it can be organized in layers, and a higher layer depends on a lower layer through abstraction. This design aims to minimize dependencies between layers, ensuring that changes to one layer will not require significant modifications to the entire system. As a result, the application's maintainability should not be a concern, provided that the system was structured and modularized appropriately. (Maintainability)

Performance is an important non-functional requirement so the system should be designed to handle a large volume of concurrent requests without slowing down or crashing. This includes optimizing database queries, minimizing response times, and ensuring efficient use of system resources. (Performance)

[Choose 4 NF for your system, describe them and explain why these NF are suitable for your implementation. ]

### Design Constraints

The system is built using the Spring framework, which is a widely used and popular framework for developing Java-based web applications. This framework offers a set of tools and libraries that help with the development of web applications, such as the ability to handle HTTP requests and responses, manage database connections, and more. On the data storage side I worked with MySQL.

My system follows the Model-View-Controller (MVC) architectural pattern, which separates the concerns of the application into three distinct components: the model (data and business logic), the view (presentation layer), and the controller (handles requests and manages communication between the model and view). This pattern ensures that the system is well-structured.

Also, within the application I used some external libraries in order to accelerate the development process and provide additional functionality. In order to map the database tables to Java and also to perform database operations in an efficient way, I used Hibernate and JPARepository. Lombok is another library which reduces the time and effort required for development by providing annotations to reduce boilerplate code in the source code.

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

## Glossary

1. Administrator: A user with special privileges who is responsible for managing the database, including adding, updating, and deleting users.

2. Customer: A user with normal privileges who is searching for a suitable worker to do a desired job/work.

3. Worker: A user with normal privileges who waits to be contacted by a customer or he can as well be searching for a customer to work for.

4. CRUD operations: Create, Read, Update, and Delete operations that can be performed on the database.

5. Model-View-Controller (MVC): A software design pattern that separates an application into three distinct components: the model, the view, and the controller.

6. Subfunction: A use case that is a part of a larger use case and helps to achieve the main goal of that use case.

7. UML Use Case Diagram: A graphical representation of the use cases in a system and their relationships with other system components.

8. User interface: The graphical user interface (GUI) through which the user interacts with the application.

[Present the noteworthy terms and their definition, format and validation rules if appropriate.]

# Deliverable 2

## Domain Model

1. Name: User

Attributes: id | firstName | lastName | email | phoneNumber | age

Associations: abstract parent class

1. Name: Customer

Attributes: workers | [User attributes]

Associations: a customer

1. Name: Worker

Attributes: job | customer | [User attributes]

Associations: a worker

## Architectural Design

### Conceptual Architecture

O imagine care conține diagramă

Descriere generată automat

The architectural style used for the system is REST (Representational State Transfer) and the architectural pattern is MVC (Model-View-Controller).

The REST architectural style is well suited for designing web-based systems that are scalable, flexible, and easy to integrate with other systems and services. REST defines a set f constraints and principles for creating stateless, client-server systems that communicate using a uniform interface.

The MVC pattern is a widely used pattern in web development that separates the application into three main components: the Model, the View and the Controller. The Model represents the data and the business logic of the application, the View represents the user interface, and the Controller handles user input and updates the Model and View accordingly.

In the case of my application, the REST architecture is used to create a set of resources that can be accessed and manipulated using HTTP requests. For example, the system has resources for tracking the progress made by users. Clients can access these resources using HTTP requests, and the server can respond with the appropriate representations of the resources.

### Package Design

O imagine care conține diagramă

Descriere generată automat

### O imagine care conține diagramă Descriere generată automatComponent and Deployment Diagram

# Deliverable 3

## Design Model

### Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

### Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

## Data Model

[Create the data model for the system.]

# System Testing

[Describe the testing methides and some test cases.]

# Future Improvements

[Present some features that apply to the application scope.]

# Conclusion

# Bibliography