**Software Engineering Process**

Version 1.0

for

SESTOPIA

Prepared by

**Team Members (Alphabetical)**

Ali Zafar Iqbal  *40076898*

Baiyu Huo  *40076004*

Chitra Gunasekaran   *...*

Dhruv Goyani  *...*

Mahsa Hemati  *...*

Michael Hanna  *40075977*

Shahryar Haghighifard  *...*

Uchechukwu Iroegbu  *...*

1. Introduction

# Purpose

The purpose of this document is to list all the functional, non-functional and other requirements of an online car renting management system. The audience of this document includes the administrator of the system and the clerks of the company. This document also provides information to the requirements team, design team, and other members of the front-end and back-end developing teams.

# Scope

This document presents the architecture of web application (MVC), its functional (for example: Reserve and Rent a vehicle) as well as non-functional requirements (for example: Security and concurrency). Different kinds of users (Clerk and Administrator) and their characteristics. It also includes use case view, Sequence Diagram and analysis models.

# Definitions, acronyms, and abbreviations

## N/A

# References

N/A

1. Overall Description

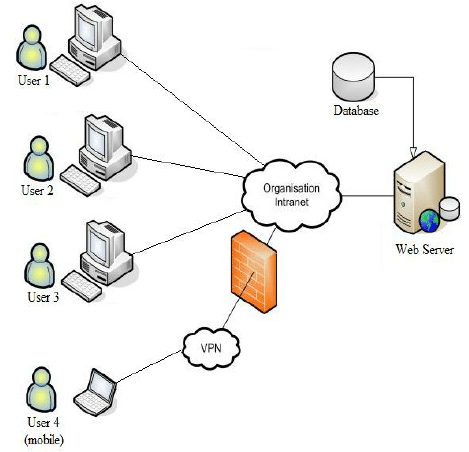


Figure 1. A client-server architecture.

# Product perspective

SESTOPIA will be an **information-intensive Web Application**. And it is **navigable and searchable** and it is accessible to anybody, anywhere.

Also, it is possible to **modify** SESTOPIA easily, and requiring small amount of time and effort by the Administrator.

# Product functions

The following are the high-level functionalities which are broken down into specific functions. Transaction management

* Navigation
* Search

# User characteristics

There are two types of users in the system.

|  |  |
| --- | --- |
| **User Type** | **Description** |
| Administrator | The administrator of the system manages:   * Add new skill * Delete skill * Modify skill   These can be managed from the server side. |
| User | The User of the system manages:   * Navigate to any skill * Search skills   It usually has simple knowledge to use the internet. It means the only skill needed by a user is the ability to browse in the web pages. |

# Constraints

This is the list of constraints that must be respected.

* The project must be completed by Aug 13.
* The developer team must maintain a relational database management system which is MySQL Database.
* The server-side application must be implemented using the Node.JS
* Fron-end must be implemented using HTML, CSS and JS

# Assumptions and dependencies

Currently, there are no assumptions or dependencies which affect this project.

# Specific requirements

## External interfaces

|  |  |
| --- | --- |
|  | |
| Screenshots of the website |  |
|  |  |
|  |  |
|  | |

# Functional requirements

## Actor / goal list

|  |  |
| --- | --- |
| Actor | Goal |
| User | 01. Navigate Skill  02. Search contents in all skills. |
| Administrator | 01. Add Skill  02. Modify Skill  03. Delete Skill |

## Use Case view

A close up of text on a white background

Description automatically generated

Figure 2. Use case model.

**Search Content in Skill :** User shall search content in skills

**Navigate through all skills :** user shall navigate through all skills.

**Add Skill :** Admin shall add skill from the server side.

**Modify Skill :** Admin shall modify skill from the server side

**Delete Skill :** Admin shall delete skill from the server side

# Non-functional requirements

# Accessibility

The web application has been deployed and it is available to anybody, anywhere, using any interactive device connected to the Internet

# Performance efficiency

appropriate architectural style that support quality characteristics like separation of concerns, modularity, reusability and adaptability.

# Compatibility

Compatibility testing will be used to ensure compatibility of the web application with various other objects such as browser.

# Usability

Designing an interface that will be easy to learn. The interface will be easy to learn and navigate.

# Maintainability

Identifying exceptions or faults, debug or isolate faults, in this project maintenance can be done during testing and overviewing iteration.

# Portability

The web application can be run on both windows and linux; and shall run on all computer web browser

# Design constraints

##### Software languages

* FrontEnd: HTML, CSS, JavaScript.
* Backend: Node.JS
* Database: MySQL.

##### Development Tools

* Visual Studio Code,…..mentioned all the tools that have been used

# Persistence

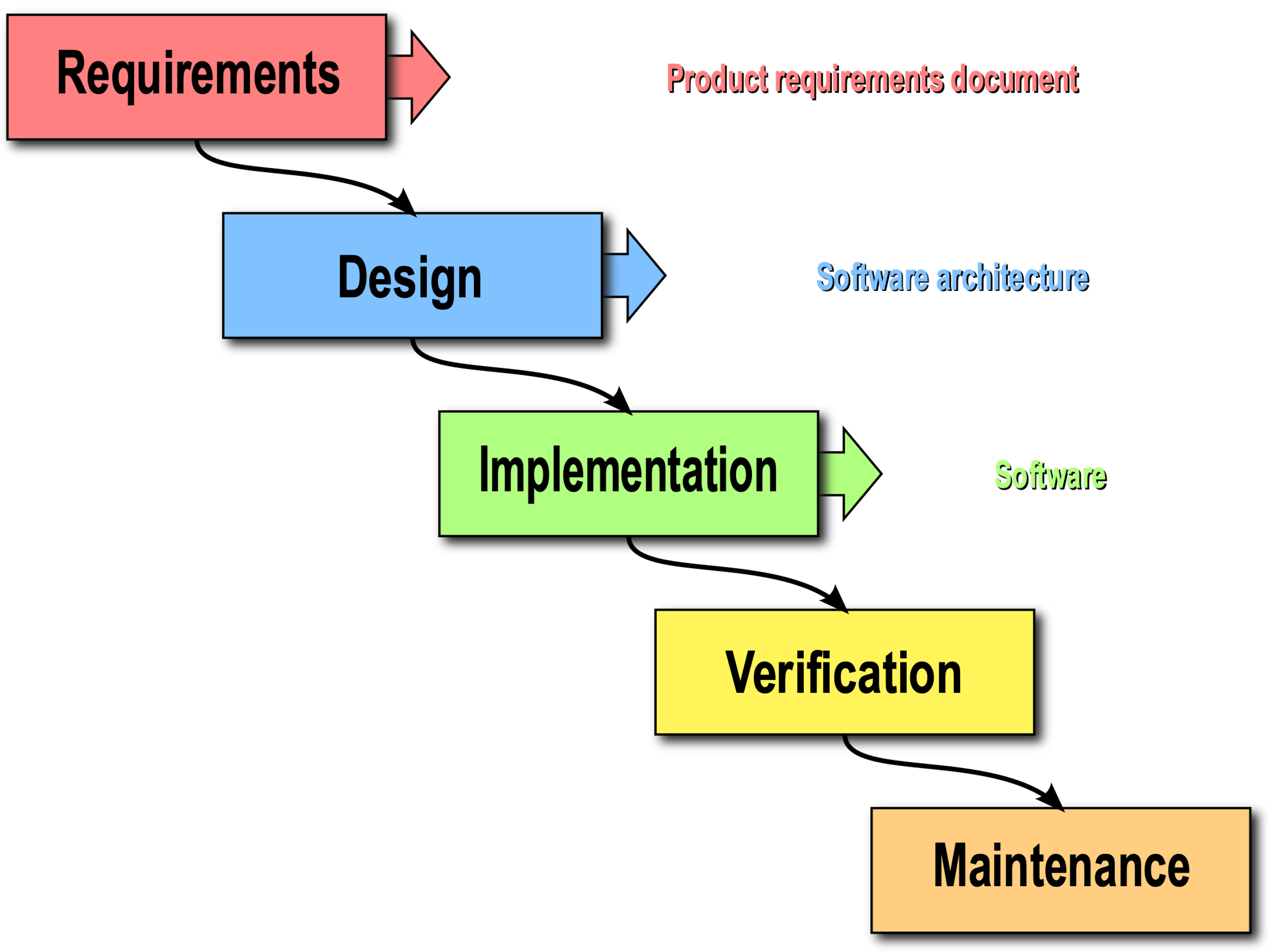
# MySQL has been used to persist the data.

# error handling

# Analysis Models

# Software Process

**Waterfall Model** has been used to ensure **low coupling** among all the phases



# Requirements

Requirements elicitation has been done through reading the project description, asking professor and TA.

# Design

# Software Architecture

A picture containing game

Description automatically generated

The use of MVC components improves **maintainability**

A decomposition of the system into three components:

* A model containing the core functionality and data,
* views displaying information to the user, and
* controllers that handle user input (Search and navigation).

Other option that can be used is to use a static website, however this approach isn’t the best practice for maintainability.

# UML DIAGRAM

#### Domain Model Diagram or class Diagram or CRC

# Front End Design

Available tools in the market

tool used

Reason to do the design like that

# Logo Design

Available tools to design logo

Tool that has been used

Reason behind doing the logo that

# Implementation

# Front End Implementation

Available Technologies in the market

Technologies used

Reason to use that Technology

# Back End Implementation

**Available Technologies in the market**

Laravel framework, Node.JS

**Technologies used**

Node.JS

**Reason behind using that technology**

# Persistence Layer

# Database Design

# Database Table

# Verification

Unit Testing

# Maintenance

The system can be maintained easily because of the software architecture that has been deployed.

There are Low coupling between layers (Model, View and Controller) and persistence data is on a separate layer.

Data can be easily maintained from the server side.

# Skills

All Skills to be added here