|  |
| --- |
| **Project Title Subtitle** |

**[Name(s) of student(s), student number, photographs]**

**Supervisor: [Name of supervisor(s)]**

**[Name and logo of educational institution]**

**[Logo of companies included]**

**[Number of characters]**

**[Study program]**

**[Semester]**

**[Date]**

**Table of content**

[Abstract iii](#_Toc18659739)

[1 Introduction 1](#_Toc18659740)

[2 Analysis 2](#_Toc18659741)

[2.1 Requirements 2](#_Toc18659742)

[2.2 Functional Requirements 2](#_Toc18659743)

[2.3 Non-Functional Requirements 3](#_Toc18659744)

[3 Design 4](#_Toc18659745)

[4 Implementation 5](#_Toc18659746)

[5 Test 6](#_Toc18659747)

[5.1 Test Specifications 6](#_Toc18659748)

[6 Results and Discussion 7](#_Toc18659749)

[7 Conclusions 8](#_Toc18659750)

[8 Project future 9](#_Toc18659751)

[9 Sources of information 10](#_Toc18659752)

[10 Appendices 1](#_Toc18659753)

# Abstract

**[Ionut]**

*The Car Rental System is a comprehensive platform designed to streamline the process of renting vehicles while ensuring a user-friendly experience. The project aims to provide an efficient and modern solution for individuals and businesses seeking to rent vehicles, facilitating the entire rental journey from booking to return.*

*Employing a three-tier architecture, the Car Rental System encompasses a range of technical choices. The user interface is crafted using Blazor App, harnessing RESTful services for seamless interactions. The logic layer is constructed with the SpringBoot Framework in Java, establishing a connection to the first tier via REST. The third tier, responsible for database management, is developed in C# using the EntityCore Framework. Communication between tiers 2 and 3 is facilitated through the utilization of gRPC services, ensuring smooth data exchange.*

*Users are empowered to effortlessly list vehicles for rental, creating rental posts with relevant details. The system enables users to explore available vehicles, make bookings, and manage reservations. Additionally, the platform supports essential functionalities such as user authentication, comments for validation, and a comprehensive blocking system to uphold a secure environment.*

*Throughout the development process, several objectives have been realized, contributing to the creation of a functional and user-centric system. While not all objectives were fully met, the Car Rental System provides users with a practical means to engage in vehicle rental transactions, fostering a marketplace for renting and renting out vehicles.*

*In essence, the Car Rental System serves as a modern and efficient solution, connecting users with diverse vehicle rental needs. The platform's multi-tier architecture, encompassing Blazor App, SpringBoot, and EntityCore, enables seamless interactions and data management, ultimately culminating in a user-friendly and functional car rental experience.*

# Introduction [Ionut]

In today's dynamic landscape, the utilization of advanced technology is becoming increasingly prevalent, reshaping traditional transactions and introducing digital avenues for fulfilling various needs. Within this context, our project takes root, striving to revolutionize the realm of car rental services through a modern and efficient approach.

The car rental industry plays a vital role in facilitating transportation needs for individuals and businesses alike. Our project aims to enhance and streamline this experience by creating a comprehensive car rental platform that caters to a wide range of users. The project's focal point revolves around offering a seamless interface where users can easily search, book, and manage their car rental reservations.

Traditionally, car rental services have been associated with cumbersome processes, limited options, and fragmented information. The lack of a unified platform often leads to inefficiencies and challenges for both customers and service providers. Recognizing this existing context, our project seeks to address these pain points by providing a user-friendly and technologically advanced solution.

The stakeholders of this project encompass a broad spectrum, including individual users seeking reliable transportation, business travelers in need of flexible options, and car rental agencies looking to enhance their customer experience. The customer, in this case, is any individual or entity looking to rent a vehicle for personal or business use.

One of the key challenges we aim to tackle is the fragmented nature of the car rental process. By creating an integrated platform that connects customers with a diverse fleet of vehicles and offers transparent pricing and booking procedures, we seek to streamline the car rental journey. This project's relevance is underscored by the increasing demand for efficient and user-centric car rental solutions that cater to the digital-savvy generation.

The objectives of our project encompass the development of a user-friendly graphical interface using Blazor App, establishment of a robust logic layer with SpringBoot Framework, and efficient data management through the EntityCore Framework. Additionally, we aim to ensure seamless interactions between tiers through gRPC services, enhancing overall system performance and responsiveness.

Delimitations within our project involve focusing solely on the car rental process itself and not incorporating payment systems into the solution. This decision is made to prioritize the core functionalities of the platform and ensure a streamlined and efficient user experience.

As we delve into the subsequent chapters of this report, we will delve deeper into the analysis, design, and implementation of the Car Rental System. By comprehensively addressing the identified challenges and leveraging advanced technologies, we anticipate presenting a well-rounded solution that transforms the car rental experience for all stakeholders involved.

# Analysis

Our Car Rental System embarks on a journey to seamlessly connect users with vehicular mobility solutions. Adopting a holistic approach, our development team delves into the project from a strategic vantage point, aiming to comprehensively understand the challenges and intricacies involved. This higher perspective ensures a deeper grasp of the car rental landscape, ultimately enhancing the quality and effectiveness of the delivered product.

A list of functional and nonfunctional criteria is provided below.

## Requirements

The Requirements section serves as a comprehensive delineation of both functional and non-functional imperatives. These requisites function as a formal agreement between the project stakeholders and serve to establish a shared comprehension of the project's objectives.

In this section, users are identified and their roles are elucidated through the lens of actor descriptions, personas, and scenarios. By adopting this approach, a clear understanding of user needs and interactions is established.

The SMART (Specific, Measurable, Achievable, Relevant, Time-bound) principle and the MoSCoW (Must have, Should have, Could have, Won't have) prioritization technique are judiciously employed to ensure the accuracy, viability, and relevance of the requirements.

The subsequent presentation entails a methodically numbered and meticulously ranked compilation of requirements, encompassing the expectations and mandates of users, customers, and stakeholders. Each requirement is formulated with precision and testability, aligning with the overarching aim of fostering a coherent and effective project framework..

## Functional Requirements

## 

## As a user, I want to create an account, so that I can access the system's features.

## As a user, I want to securely log in, using my credentials.

## As a user, I want my password to be stored securely, protecting my sensitive data.

## As a user, I want to search for available cars, based on location, dates, and vehicle preferences.

## As a user, I want to filter search results, to refine my choices.

## As a user, I want to easily book a car, for a specific duration.

## As an admin/staff, I want to manage the car inventory, adding new cars, updating availability, and removing outdated listings.

## As a user, I want a smooth rental process, with efficient collection of necessary information.

## As a user, I want to pay for rentals using a secure method, integrated into the system.

## As a user, I want to receive transaction documentation, including invoices and receipts.

## As a user, I want to view and manage my reservations, with the ability to modify or cancel bookings.

## As a manager, I want access to comprehensive rental reports, including activities, revenue, and inventory status.

## As a user, I want to create posts to showcase my products or services.

## As a user, I want to categorize my posts, ensuring organized presentation.

## As a user, I want to view contact details of post owners, facilitating communication.

## As a user, I want to bookmark posts, for future reference.

## As a user, I want to leave comments on posts, enabling communication with sellers.

## As a user, I want to manage and update my posts.

## As an admin, I want to manage reported posts, ensuring platform safety.

## As an admin, I want to block user accounts, when necessary for security.

## As an admin, I want to unblock user accounts, as needed.

## As a user, I want to rate other users, evaluating their reliability.

## As a user, I want to update my account information, keeping it current.

## As a user, I want to view my list of posts, for easy tracking.

## As a user, I want to manage my bookmarked posts, ensuring an organized collection.

## Non-Functional Requirements

* 1. As an user, I desire a user-friendly interface that enables easy navigation within the system.

Use case diagram

Use case description

Activity Diagram

Domain Model

Security Analysis

Risk assesement Model

# Design

The design phase establishes the architecture of the system and transforms the analysis artifacts into practical and executable models.

## Architecture

Image

The widely adopted software architecture for conventional client-server applications is the three-tier architecture, dividing applications into three distinct computing tiers—both logically and physically (Brooks, 2022). Selected for its scalability, reliability, and security attributes, this architecture also expedited development by enabling parallel work on each tier by different team members.

As depicted in the diagram, the presentation tier, application/logic tier, and data tier were realized using Blazor (C#), SpringBoot (Java), and EFC (C#) correspondingly.

## Choice of Technologies

### Presentation Tier

This is the point of interaction between the system and its end users. Blazor was selected to craft the user interface for the client due to its ability to harness the benefits of a sophisticated, contemporary single-page application (SPA) using user-friendly languages like HTML, CSS, and C#.

This tier communicates with the application tier by sending HTTP requests to RESTful web services. Objects are serialized and deserialized into JSON format to facilitate the transmission of messages with the application tier.

### Application Tier

This tier encapsulates all of the application logic. It accepts messages from the presentation tier and handles the forwarding and retrieval of data from the data tier as required. The Spring Boot framework was opted for in this tier due to its incorporation of dependency injection, enabling a loose coupling throughout the application. The framework's auto-configuration also served as a significant advantage.

For communication with the data tier, Google's gRPC service is employed. By default, gRPC employs Protocol Buffers, Google's well-established open-source method for serializing structured data (grpc.io). This ensures effective communication between servers.

### Data Tier

The data tier assumes the responsibility of storing, retrieving, updating, and deleting all system data. The entirety of the data is stored within a PostgreSQL database. The database was developed iteratively, utilizing Microsoft's Entity Core Framework (EFC). Entity Framework Core operates as an object-relational mapper (ORM), establishing an intermediary layer between the domain model implemented in code and the database itself. EF Core serves as a data access API, facilitating interaction with the database via .NET POCOs (Plain Old CLR Objects) and strongly typed LINQ (Microsoft). A comprehensive account of the database construction is presented in the subsequent section.

As previously mentioned, communication with the application tier is facilitated through GRPC services.

## Entity Relationship (ER) Diagram

Image

The entity relationship diagram was generated based on the domain model established during the analysis phase. An Entity Relationship (ER) Diagram serves as a specific type of flowchart, portraying the interconnections among "entities" like individuals, objects, or concepts within a given system. ("ER Diagram (ERD) - Definition & Overview | Lucidchart")

The depicted diagram illustrates the interconnections among the entities within the system.

## C#- Class Modal Diagram

Image

This diagram is a result of the preceding ER diagram. It showcases the formulated classes that facilitate EFC in creating connections and their associations within the PostgreSQL database. Additionally, this diagram elucidates the manner in which data is stored in the database.

## Component Structure

**Image**

The component structure elucidates the interdependence among the various components. As depicted in the illustration, the presentation tier's blazorApp registers the services within the contracts components, which are then realized by the HttpClients responsible for executing HTTP requests to the RESTful services situated at the application tier. These messages are transmitted in JSON format through the serialization of objects.

Upon converting the JSON back into objects through deserialization, the controllers in the second-tier will relay these messages to the application classes housing the core business logic. The client interfaces are then engaged to manipulate or retrieve the requested data as per the directives of the application classes. Utilizing protocol buffers files, the client interfaces of GRPCclients undertake the serialization and deserialization of messages towards the data layer. Here, EFC establishes a connection with the database to effectuate any alterations to the data.

## Class Diagram

Image

Serving as a guiding template for the system's realization, the class diagram functions as a foundational framework. Its primary objective lies in modeling the static perspective of an application. Among all diagrams, class diagrams stand out as the ones most directly aligned with object-oriented languages, rendering them extensively employed during the construction phase. ("UML - Class Diagram, TutorialSprint").

Due to the comprehensiveness of the overall class diagram, which might necessitate considerable time for complete comprehension, a representative class diagram has been selected here to exemplify the process of adding a car. This specific functionality, crucial to the system, was among the initial features to be developed. The patterns of flow and interconnections between classes across different tiers remain largely consistent across other functionalities. For a comprehensive class diagram of the entire project, please refer to Appendix - X.

In the presented CarCreation page, there is an awareness of the ICarService interface, provided by contracts and implemented by CarHttpClient within HttpServices. To maintain the principles of dependency injection, these interfaces are registered as services and are injected as needed, fostering a loosely connected system architecture.

CarHttpClient is responsible for the serialization of messages into JSON format and for issuing HTTP requests to the RESTful APIs of the CarController within the application tier. Upon deserialization, these messages are conveyed to the CarService interface through the controllers. The implementation of this interface encompasses the application's logic, potentially involving data persistence and necessitating interaction with CarClient. In turn, CarGRPCClient encodes these messages into binary format and dispatches them to the data tier using Protocol Buffers.

Within the data tier, CarProtoImpl inherits the Car structure. Messages originating from the application tier find their destination in Carbase, an entity auto-generated from the proto files. Ultimately, these messages traverse to the EFC via ICarService in contracts, where CarDAOImpl employs the DbAccess class to access PostgreSQL for storage and retrieval.

### SOLID Principles

The project adhered closely to robust design principles, ensuring that each class maintained a clear and singular responsibility, minimizing the risk of future alterations affecting unrelated areas. While the Lisskov-substitution principle held lesser significance due to the absence of inheritance within the system's structure, the adherence to the dependency inversion principle was paramount. All modules were intentionally reliant on interfaces rather than implementations, as depicted in the class diagram.

Both Blazor's service registration functionality and the Spring framework inherently employed dependency injection, contributing significantly to the preservation of a loosely connected system. This design approach safeguarded the system's scalability and facilitated seamless expansion while mitigating the impact of changes on other components.

## Sequence Diagram

IMAGE

A sequence diagram falls under the category of interaction diagrams as it illustrates the coordinated actions and sequencing of operations among a set of objects. These visual representations serve as valuable tools for software developers and business practitioners, aiding in the comprehension of system requirements or the documentation of established processes. Often referred to as event diagrams or event scenarios, sequence diagrams offer a succinct depiction of dynamic interactions within a system.

Presented above is the sequence diagram detailing the process of adding a post. This diagram elucidates the sequence of method calls occurring across various classes situated in different tiers of the architecture.

## Security

The ensuing strategies outline the recommended approaches for addressing security threats:

|  |  |
| --- | --- |
| Threat | Mechanism Suggested |
| Elevation of privilages | Implement strong role-based access control (RBAC) mechanisms. Define different user roles with appropriate permissions and ensure that access is granted only based on authenticated roles. |
| SQL Injection | Utilize parameterized queries and prepared statements to validate and sanitize user inputs. Employ web application firewalls (WAFs) to detect and block SQL injection attempts. |
| Cross-Site Scripting (XSS) | Implement input validation and output encoding. Use security libraries and frameworks to automatically sanitize user inputs and prevent the execution of malicious scripts. |
| Data Breach | Implement encryption for data at rest and during transmission. Regularly monitor and audit access logs, and employ intrusion detection systems (IDS) to identify unusual activities. |
| Distributed Denial of Service (DDoS) Attacks | Use rate limiting, traffic filtering, and load balancing to mitigate DDoS attacks. Employ cloud-based DDoS protection services for additional defense against large-scale attacks. |

# Implementation

Adhering to the iterative sprint approach, the team transitioned from conceptualizing the design to the practical implementation of the code. A pivotal element of the software involves enabling users to create and list items for sale. This section will delve into the process of achieving this essential functionality.

## Client

Leveraging the capabilities of Blazor, a .NET web application, we are crafting the user interface as detailed in the design phase. Blazor provides a comprehensive framework for web application development, encompassing essential features such as navigation and index pages. The team's approach involves harnessing the power of HTML, CSS, and Bootstrap to enhance the user experience, thereby making the interface more intuitive and engaging.

In order to create an order, the user is required to be logged in with a standard user account. The process of authentication and authorization is facilitated through ASP.NET's attribute mechanism ([Authorize]), ensuring user validation. Consequently, if the user is not authenticated, the page remains inaccessible, providing an added layer of security.

Upon successful user validation, it is mandatory for them to complete all mandatory fields before triggering the invocation of the CreateBooking method.

Image

Within a try-catch procedure, each field undergoes thorough examination. In cases of errors or if the file size surpasses a predetermined limit, an Exception is triggered.

The SOLID Principle is diligently followed in the codebase. The IBookingService interface governs tasks pertaining to posts, including the addition of new posts. Should a file be uploaded, it is first converted to MultiPartFormDataContent before being dispatched. Assuming all prerequisites are met, the AddBooking function within BookingHttpClient is invoked.

Image  
  
  
In this procedure, the post is serialized and forwarded as JSON to the application tier using the REST-API's post method on the designated URI. To ensure security, a header is required before establishing a connection to the API – a topic to be expounded upon subsequently. If a file has been uploaded by the user, upon successful post addition, the transformed file, accompanied by the post ID in the path, is transmitted to the application tier.

## Application Tier

As elucidated in the design segment, the application tier is constructed using Java and SpringBoot, with Lombok being among the libraries employed to enhance coding efficiency and swiftness. Operating as a server for the Client layer through the RestApi (HTTP protocol), and as a client for the Database layer via gRPC, this tier undertakes the responsibility of acquiring data from the client. Subsequently, the service component is tasked with retrieving the item via HTTP, facilitating its processing, and further transmitting it to the database layer utilizing gRPC.

Image  
  
  
Explanation of the Image

## Database

The database tier was set up to manage objects, handling their storage and retrieval from the database. This design was chosen due to the software's deployment of primary functions across separate servers, eschewing a monolithic approach. In achieving this, the database layer was structured with the assistance of gRPC, and for the task, Entity Framework Core (EFC) emerged as a fitting solution. Consequently, a combination of C# language and ASP.NET was employed.

Image

Explanation

A screen shot of a computer code

Description automatically generated

For the establishment of a database table and the connection of the program to a remote PostgreSQL database, a Context file is developed, implementing the DbContext offered by Entity Framework Core (EFC). This file encompasses the assignment of tables, along with their primary and foreign keys. Furthermore, it serves as the point of connection to the remote database.

## Security

During the implementation phase, the team has integrated security measures into the RESTful web server, incorporating the Bcrypt password-hashing algorithm. A dedicated API key has been set up for the RESTful server, and robust HTTP security protocols have been implemented across all server endpoints.

Image

Image

# Test

## The primary aim of this segment is to provide the reader with insights into the accomplishments of the development team. Readers will have the chance to explore the executed user stories and assess potential areas for further enhancements.

# Results and Discussion

|  |  |  |
| --- | --- | --- |
| As a user, I want to be able to create an account easily. | Implemented | Test Successful |
| As a user, I want to browse through a list of available cars. | Implemented | Test Successful |
| As a user, I want to see detailed information about each car's features. | Implemented | Test Successful |
| As a user, I want to select a car and book it for specific dates. | Implemented | Test Successful |
| As a user, I want to receive a confirmation after booking a car. | Implemented | Test Successful |
| As a user, I want to view my booked cars and their details. | Implemented | Test Successful |
| As a user, I want to cancel a car booking. | Implemented | Test Successful |
| As a user, I want to see rental rates and compare options. | Implemented | Test Successful |

|  |
| --- |
| Implemented User Stories |

|  |
| --- |
| Could Have |

|  |  |  |
| --- | --- | --- |
| As a user, I could have the option to filter search results by specific car features, such as air conditioning, GPS, or automatic transmission, to find a vehicle that meets my preferences. | Not Implemented | Not Implemented |
| As a user, I could have the ability to rate and leave feedback about my rental experience, helping other potential renters make informed decisions. | Not Implemented | Not Implemented |
| As a user, I could have the option to receive notifications and updates about my upcoming reservations, reminders for returning the car, and any changes to my booking. | Not Implemented | Not Implemented |
| As a user, I could have the choice to view detailed vehicle specifications, including fuel efficiency, engine capacity, and safety features, to make well-informed rental decisions. | Not Implemented | Not Implemented |
| As a user, I could have the ability to extend my current car rental reservation if I need the vehicle for a longer period than initially booked. | Not Implemented | Not Implemented |
| As a user, I could have the option to select and purchase additional insurance coverage during the booking process to ensure added protection during the rental period. | Not Implemented | Not Implemented |
| As a user, I could have the ability to search for nearby gas stations and receive directions through integrated maps, making it easier to refuel before returning the car. | Not Implemented | Not Implemented |

As observed in the tables above, the project team has successfully realized the majority of the user stories, acknowledging the presence of some challenges along the way. Despite these challenges, users can now create accounts, post products or services, and explore the platform through existing listings.

With this achievement in mind, the primary goal has been attained, although further opportunities for enhancement remain.

# Conclusions

In the swiftly evolving landscape of technology, remaining competitive amidst rapid innovations poses a significant challenge. The primary goal of our team has been to establish a platform for seamless car rental experiences. This objective has been realized through the implementation of a robust three-tier architecture, encompassing the client, application, and database components. This dynamic system accommodates distinct user roles, each with varying access privileges.

Commencing with a thorough exploration of the problem domain, we derived specific requirements that laid the foundation for developing comprehensive use cases and a detailed domain model. This domain model, in turn, served as the cornerstone for creating intricate class diagrams during the design phase. Subsequently, these conceptual designs were seamlessly integrated into the presentation, application, and data tiers, facilitated by RESTful services, HTTP calls, and GRPC communication protocols for efficient data interchange. The utilization of Entity Framework Core (EFC) in conjunction with C# streamlined interactions with the PostgreSQL database, empowering users to engage in a range of activities, including car searches, bookings, and management.

While the application layer operates discreetly behind the scenes, it holds a pivotal role as the nerve center for logic processing. Equally essential, the database layer acts as the repository for all critical data, underpinning the core functionality of the entire project. Whether users are searching for available cars or making bookings, data seamlessly traverses between the client tier and this foundational layer.

Given these achievements, it is fitting to conclude that the project has admirably achieved its primary objectives, firmly establishing itself as a resounding success in the realm of modern car rental systems.

# Project future

Due to constraints on time, inadequate team resources, and insufficient communication, the system has been developed with simplicity, foregoing potential advanced features. It's worth noting that the absence of futuristic elements is not a reflection of the team's capability, but rather a result of the circumstances at hand.

Presently, in order to purchase items or services from the platform, users are required to either arrange physical meetings or resort to third-party payment systems. However, these approaches come with drawbacks. Cash transactions, while prevalent in some regions, have proven ineffective in modern contexts like Denmark, where digital payments are progressively favored (Brushan 2017).

Conversely, the cumbersome process associated with third-party payment systems has been observed to significantly impact customer satisfaction. Instances of users abandoning orders and discontinuing their engagement with the platform due to this inconvenience are not uncommon (Konior 2016).

Acknowledging these insights, the team recognizes the importance of incorporating a streamlined payment method as a strategic move to enhance the platform's appeal. Moreover, the user interface, which serves as the gateway to the system, plays a pivotal role in the overall user experience. A more intuitive and user-friendly interface tends to result in higher user engagement.

Given these considerations, the team envisions the implementation of a comprehensive payment system and an intensified focus on refining the frontend of the system as the logical next steps. While the current system may lack certain advanced features, it serves as a foundation that can be built upon to create a more compelling and user-centric platform.

# Sources of information

Brooks, G., 2022. "What is Blazor, and why should you use it? - Fabrity Software House." Available at: https://fabrity.com/blog/software-engineering/what-is-blazor-and-why-should-you-use-it/#:~:text=Let's%20consider%20the%20advantages%20of,work%2C%20allows%20for%20faster%20development. [Accessed 25 May 2022].

CamSoper. "Understanding EF Core - Learn | Microsoft Docs." Available at: https://docs.microsoft.com/en-us/learn/modules/persist-data-ef-core/2-understanding-ef-core. [Accessed 25 May 2022].

"ER Diagram (ERD) - Definition & Overview | Lucidchart." Available at: https://www.lucidchart.com/pages/er-diagrams. [Accessed 25 May 2022].

"UML - Class Diagram." Tutorialspoint. Available at: https://www.tutorialspoint.com/uml/uml\_class\_diagram.htm. [Accessed 26 May 2022].

"UML Sequence Diagram Tutorial | Lucidchart." Available at: https://www.lucidchart.com/pages/uml-sequence-diagram. [Accessed 26 May 2022].

Business 2 Community. "How Payments With Redirection Affect Your Business." Available at: www.business2community.com, 2016.

Cards International. "Denmark gets closer to cashless - Cards International." Available at: https://www.cardsinternational.com/country-surveys/denmark-country-report-cards-cashless/ [26.5.2022].

TheFreeDictionary.com. "One man’s trash is another man’s treasure - Idioms by The Free Dictionary." Available at: https://idioms.thefreedictionary.com/one+man%27s+trash+is+another+man%27s+treasure [27.5.2022].

Wikipedia. "Bcrypt - Wikipedia." Available at: https://en.wikipedia.org/wiki/Bcrypt [Accessed 28 May 2022].

Bacon, M. 2021, "What is Security?," SearchSecurity, www.techtarget.com, viewed 31 May 2022, https://www.techtarget.com/searchsecurity/definition/security#:~:text=Security%20for%20information%20technology%20(IT,otherwise%20known%20as%20threat%20actors..

Hewko, A. 2021, "STRIDE Threat Modeling | What Is It? | Explanation and Examples | Read," Software Secured, www.softwaresecured.com, viewed 31 May 2022, https://www.softwaresecured.com/stride-threat-modeling/.

"What is Security Risk Assessment and How Does It Work? | Synopsys" n.d., What is Security Risk Assessment and How Does It Work? | Synopsys.

# Appendices

1. Appendix A ➔ Source code (GITHUB LINK)

2. Appenix B ➔ Use case diagram and descriptions

3. Appendix C ➔ Project description

4. Appendix D ➔ Domain Model

5. Appendix E➔Activity Diagrams and System sequence diagram

6. Appendix F ➔ User guide

7. Appendix G ➔Class diagram

**Process Report**

Group 1

Ionut Adrian Rotaru (304491)

Iustin Gricorcea (305546)

Supervisor: Troels Mortensen

**Software Technology Engineering**

9 of August 2023

**Table of Contents**

1. Introduction
2. Group Description
3. Project Initiation
4. Project Description
5. Project Execution
6. Personal Reflections
7. Supervision
8. Conclusions
9. **Introduction**

This report outlines the strategies and endeavors undertaken during the third-semester project, encompassing the hurdles faced and the strategies pursued.

Given that the team remained consistent from the preceding semester, the members were well-acquainted with each other's work methodologies. The initiative commenced with formulating a comprehensive agreement that delineated the group's anticipations. Subsequently, two concepts were conceived to serve as catalysts for fulfilling the requisites. Eventually, in collaboration with supervisors, the decision was made to introduce a mechanism facilitating the connection between individuals desiring to vend goods or services, thereby eliminating conventional advertisements like "I am selling wood pallets" from stores.

Over the course of the semester, we engaged in three distinct courses that significantly contributed to our project objectives and requirement implementation. During the DNP course, we gained insights into developing client/server web pages, inter-component communication, and data storage techniques, including subsequent transfer to a database. The SDJ3 class equipped us with Java language proficiency and knowledge of various middleware such as Spring and RabbitMQ.

Furthermore, the NES training augmented our understanding of cybersecurity aspects, encompassing diverse attack methodologies and effective preventive measures. Our collaborative efforts initially entailed weekly meetings, which evolved into daily sessions as the project deadline approached. These sessions were crucial for addressing code development challenges, refining sprint plans, and strategizing subsequent steps. Communication was streamlined using platforms like Microsoft Teams and Messenger, while JIRA was employed as a centralized hub for workload management and sprint tracking. For code management, the source code repository is available on GitHub.

1. **Group Description**

The collective composition of our team comprises two individuals of Romanian nationality, an aspect that has precluded the viable application of the Hofstede approach to illustrate variances in cultural attributes inherent to disparate nations. In the course of our ongoing project endeavor, a robust framework for inter-team communication was meticulously cultivated, and the apportionment of tasks among all team constituents was executed with a discernible equitability, thereby conferring each member with a commensurate degree of authority reminiscent of a scrum-master.

This judicious allocation of tasks was tailored to encompass a comprehensive spectrum of responsibilities, spanning the entire trajectory of user story conception, evolution, and eventual culmination. In addition to this, our inquisitive purview was extended towards the delineation of individualism. The outcomes borne of this investigation, strikingly congruent in nature, vividly underscored a shared proclivity within our team towards cohesive participation and productive interaction within the precincts of collaborative group dynamics.

The final dimension of our exploration encompassed the facet of masculinity. A salient observation emerged from our collective operational ethos, as both members demonstrated a protracted history of fruitful collaboration, thereby reinforcing a collective ethos underscored by principles of parity and unity. This substantial collaborative ethos, in turn, accentuates our mutual conviction in the potential efficacy of negotiation and compromise as preeminent mechanisms for the resolution of any potential disagreements that may arise.

1. **Project Initiation**

We kicked off our journey with the SEP3 project re-exam right at the start of July. Our heads together with our awesome supervisor led us to this exciting idea of building a web app that could bring folks together to sell their stuff and services. But of course, there were twists and turns to navigate along the way.

Choosing the tech stack was like picking the perfect ingredients for a recipe. After a bunch of brainstorming sessions and a deep dive into research, we settled on C# and Blazor for our Tier 1, which is like the front-end show. And then, weaving everything together, we had Java SpringBoot for Tier 2, where all the backstage magic happens. The bridge between Tier 2 and the solid ground of Tier 3, built with C# Entity Framework, was beautifully crafted using GRPc.

You know, we had this agile mindset from the get-go, thanks to previous experiences. We knew the road ahead wasn't all smooth sailing. Challenges were bound to pop up during the actual building phase. And boy, they did! But guess what? We didn't let that dampen our spirits.

Talking about challenges, it's like we were juggling a trio of tricky balls: time, team size, and communication. Yep, we felt the pressure of deadlines and wished we had a few more hands on deck. And, oh, the communication waves weren't always crystal clear, which added an extra layer of complexity.

So, we rolled up our sleeves, armed ourselves with JIRA – a trusty tool we were familiar with – and decided to face the storm head-on. We believed that as we went along, we'd get better at handling things. And you know what? We did! This choice felt like a warm hug of confidence in our teamwork.

Our story is one of determination, growth, and yes, a few hiccups. But that's what makes it real. With every challenge, we learned, evolved, and embraced the power of working together. And as we wrap up this chapter, we know we've grown stronger and wiser.

1. **Project Description**

As we embarked on our car rental journey, the excitement was palpable. With our toolbox of tools and tech choices ready, we set out to give life to our project. It was like revving up the engine, feeling the adrenaline rush as we hit the road.

But before we could hit the gas, we paused to ponder: Why are we doing this? Could we really pull it off? It was like looking at a map, tracing the route, and wondering if we could reach our destination. We put our heads together, connecting the dots, and the big picture started to emerge.

We dived deep into the car rental world, exploring how things worked. It was like peeling back layers to uncover what was really going on. We had our own ideas, but to put things in perspective, we needed an introduction, a doorway into the challenge we were tackling. This was the birth of our problem space, a place where we aimed to make a difference for those wanting to rent out their cars.

With visions swirling, we faced a reality check. Not all dreams could fit within our project's trunk. So, we sifted through our ideas, like sorting out what goes into the luggage and what stays behind. Our supervisors played the role of travel advisors, helping us pick the essentials.

In the shadows of ambition, the ghost of risks loomed. We navigated this territory, like sailors plotting their course to avoid rough waters. It was about being prepared for the unexpected.

The gears shifted as we stood at a crossroads. The project was a massive highway stretching ahead, and we needed a roadmap. With a year of teamwork in our rearview mirror, we decided on Scrum, a method that felt like a comfortable pair of driving gloves. And to make the most of our ride, we plotted out our journey on a timeline.

With the scene set, our project description came to life. It was like painting a picture, adding colors and details. We crafted and re-crafted, like sculptors shaping a masterpiece, all while our supervisor acted as the guide, making sure our creation stayed true to its roots.

As we entered the world of analysis, it felt like stepping onto a new road. Our direction was clear, our goals were in focus, and we were on the move. Just like a car engine humming, we were off to a promising start.

1. **Project Execution**

We rolled into our car rental project, fueled by the whispers of our previous trysts with Scrum and the unified process. Like friends reuniting after a hiatus, we decided to dance with these familiar companions once again. The idea of having a designated product owner got lost in our camaraderie; we all huddled together, recognizing that our collective journey was a symphony of shared efforts. Each sprint unfurled like a kaleidoscope, with the role of the scrum master pirouetting to a new dancer every time, ensuring that everyone took a turn in the spotlight.

Our tale began with the ink of requirements and the parchment of user stories, meticulously crafting the foundation of our project. These initial strokes were the opening chords of a melody that would soon crescendo into the cadence of our product backlogs. Assigning priority to each task felt like aligning stars in a constellation, each shining in its unique brilliance within the system's galaxy.

Short sprints became our artistic canvas, where we painted our ideas with bold strokes of creativity. These brief intervals allowed us to unveil our mistakes in real-time, like revealing hidden treasures just below the surface. Yet, in the realm of classrooms and books, our sprints spread their wings across a week, a slower rhythm that hummed in harmony with our academic responsibilities.

JIRA, our trusty ally from days gone by, stood as our guiding star in this voyage. Like a conductor leading a grand orchestra, it helped us harmonize and orchestrate our progress. GitHub, the tapestry of our digitalworld, wove us all together in a seamless embrace. We stitched together JIRA and GitHub, creating a bridge between our tasks and accomplishments, painting a vivid picture of our collaborative masterpiece.

The Unified Process guided us like a compass, steering us through the wilderness of ideas. We crafted user stories from the raw material of our project description, shaping the contours of our requirements and giving life to the tapestry of use cases. Elaboration unfurled its wings, breathing life into use case descriptions and summoning a domain model from the depths of imagination. Our journey was a dance of iterations, each sprint breathing new life into our evolving creation.

Amidst the construction phase, our vision took shape like a sculpture emerging from ablock of marble. The rhythm of oursprints sometimesstumbled over missteps, a reflection of our lively discussions and spirited debates. Beneath the surface, a conviction thrived - our project blossomed with promise, like a garden that could have flourished even more given the gift of time.

* 1. **Product Backlog**

We forged our backlogs from the essence of the project description, sculpting them into existence, and then carefully arranged them in a symphony of priorities.

1. As a user, I want to easily create an account, so I can start renting cars hassle-free.
2. As a user, I want to effortlessly browse through a comprehensive list of available cars, ensuring I have a variety of options.
3. As a user, I want to access detailed and comprehensive information about each car's features, helping me make an informed decision.
4. As a user, I want the ability to select a car from the list and smoothly book it for specific dates, streamlining the reservation process.
5. As a user, I want prompt confirmation and assurance after successfully booking a car, giving me peace of mind.
6. As a user, I want a clear and organized overview of my booked cars along with their details, allowing me to manage my reservations effectively.
7. As a user, I want the flexibility to cancel a car booking if my plans change, ensuring a user-friendly experience.
8. As a user, I want to easily access and compare rental rates across different cars, enabling me to choose the best option for my needs.
9. As a user, I want to seamlessly check the availability of cars for specific dates, preventing any potential booking conflicts.
10. As a user, I want the option to provide feedback and rate my rental experience, contributing to the improvement of the service.

Scrum to be added.

1. **Personal Reflections**

**Ionut Adrian Rotaru**

As a team, at the project's outset, we formulated a contract outlining guidelines to achieve our proposed work. These guidelines aimed to enhance communication, task comprehension, and meeting deadlines. Unfortunately, some rules were disregarded, impacting our group's productivity. Consequently, our end product lacked the expected organization and functionality. Throughout this project, I felt a deep commitment to contributing my best, which positively influenced our workload. Although transitioning from a Python background to working with Java and C# presented challenges, I enthusiastically tackled these new realms, ensuring precise task comprehension and timely completion.

Our contract not only established rules but also provided a structured framework for addressing unfinished assignments or missed meetings. This facilitated smoother communication and teamwork. Addressing minor conflicts and aligning our interests helped us better understand our tasks. In the future, I recognize the value of incorporating consequences for task delays or inaccuracies within our contract.

Despite minor challenges, our grasp of tasks and effective teamwork minimized major hurdles during system development. However, time constraints and limited supervision led to some code errors. As our sprints progressed, daily scrum meetings with our team's scrum master played a vital role.

Within this collaborative project, I observed that, while belonging to a single culture, we shared a cohesive lifestyle and approach to challenges. This unity fostered a shared sense of happiness and a consistent method for managing difficulties.

Throughout this venture, I encountered challenges while implementing the three-tier architecture system. These hurdles piqued my interest and fueled my curiosity for this field. In conclusion, my determination thrived as I learned the value of collaborative teamwork, synchronized task completion, and collective problem-solving. Looking ahead, I am motivated to once again contribute my best in the upcoming semester, striving to enhance our collaborative efforts and promptly resolve any challenges we encounter.

**Iustin Gricorcea**

1. **Supervision**

For the ongoing project, our team was fortunate to have the guidance of an assigned supervisor. His pivotal role was instrumental in both the group's establishment and the project's inception.

While consistent supervision was absent throughout the project, depriving us of ongoing feedback, we did receive invaluable insights at the project's outset. The supervisor played a pivotal role in imparting crucial information, especially during the project's initiation. This included presenting us with the project's requirements and furnishing us with a comprehensive step-by-step roadmap for optimal project execution. This collection of "how to..." insights encompassed both formal requirements and system development guidelines, proving to be a valuable compass for our team during the project's development phase.

The extent of our success in adhering to this guide will only be unveiled during the final feedback assessment.

1. **Conclusions**

Throughout this project journey, we noticed that our group's cohesiveness was a bit lower than what we usually experience in regular projects. But we kind of expected this because our project timeline fell right in the middle of our summer break. While we managed to handle it reasonably well, we all felt that there could have been better communication among us.

We did have regular meetings, but honestly, the quality of those meetings and the information we shared during them left something to be desired. Agile methodologies definitely helped us navigate this challenge to some extent. However, looking ahead, we think having a slightly heavier workload might actually help us concentrate better. We found that the meetings, though consistent, ended up feeling a bit frustrating because we didn't see as much progress as we hoped due to the limited time we had for the project.

Another factor that affected our productivity was the lack of in-person interactions. Sadly, we only managed to have face-to-face meetings at the beginning of the project. This lack of physical interaction definitely played a role in slowing things down for us. As we move forward, we're keeping all these lessons in mind and hoping to make our next project experience smoother and more productive.