Giadinhtiendung e-commerce website

Software architect document

Version 0.1

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 30/11/2023 | 0.1 | Initial writings | Lê Quang Trường |
| 02/11/2023 | 0.1 | Complete writing | Nguyễn Văn Tuấn Kiệt |
| 02/11/2023 | 0.1 | Complete writing | Bùi Nguyên Hanh |
| 02/11/2023 | 0.1 | Complete writing | Trần Ngọc Trường Thịnh |
| 02/11/2023 | 0.1 | Complete writing | Nguyễn Gia Khánh |
| 02/11/2023 | 0.1 | Complete writing | Lê Quang Trường |

Table of Contents

1. Introduction 4

2. Architectural Goals and Constraints 4

3. Use-Case Model 4

4. Logical View 4

5. Deployment 4

6. Implementation View 4

Software Architecture Document

# Introduction

Giadinhtiendung web based application, providing an online trading platform for vendors and users to sell and buy household products in a fast and convenient way.

Purpose:

* This document serves as a comprehensive guide to the architecture of the Giadinhtiendung system. Delving into various aspects of the architecture.

Scope:

* The scope of this SAD is to explain the architecture of the system. This document describes many aspects of the system architecture,in which each element takes an important role in constructing a stable system..

Definitions, Acronyms and Abbreviations:

* OOP: Object oriented programming

Overview:

* This document provides a deep understanding of the system architecture. Future sections cover the details of the 3 main views: (logical, deployment and implementation).

# Architectural Goals and Constraints

* **Server side:** The website database is hosted using Plannetscale. All communication with clients has to comply with public HTTPS,TCP/IP communication protocol standards. The website is a web based application so that it can be accessed using any regular browser such as Chrome, Microsoft Edge,...
* **Client side:** Users will be able to access ‘Giadinhtiendung e-commerce website’ only online. Clients/users are expected to use a modern web browser such as Edge, Google Chrome to get full user experience
* **Security:** On both client and server side, security is handled by NextJs built-in protection features, database connection is protected by prismadb.
* **Persistence:** Data will be stored in a central database, hosted on Plannetscale. The database is designed to meet the BCNF standard, assure minimal data duplication. All transactions will be handled with high consistency automatically by using the Prisma interface.
* **Reliability/ Availability:** System will have many undergone tests every development sprint to make sure that this will show strong reliability in practice. Furthermore, the server is designed to handle all requests separately and retrieve data from the server with well handled concurrent transactions.
* **Performance:** The system responds to any request, mainly fetching data from the database. System performance depends on the hardware specifications, network connectivity on the client side. The server’s performance is ensured by Vercel and Plannetscale scaling abilities. However, real performance can only be determined after testing and deploying the system.
* **Portability and Reuse:** In order to maintain code reusability, all functional components are well structured and organized, OOP standards are also strictly adhered, combining with NextJs well-known compatibility with various deployment services.
* **Development tools:**

Programming: Visual Code IDE, Github

Api and Front-end: Nextjs, with several utility packages.

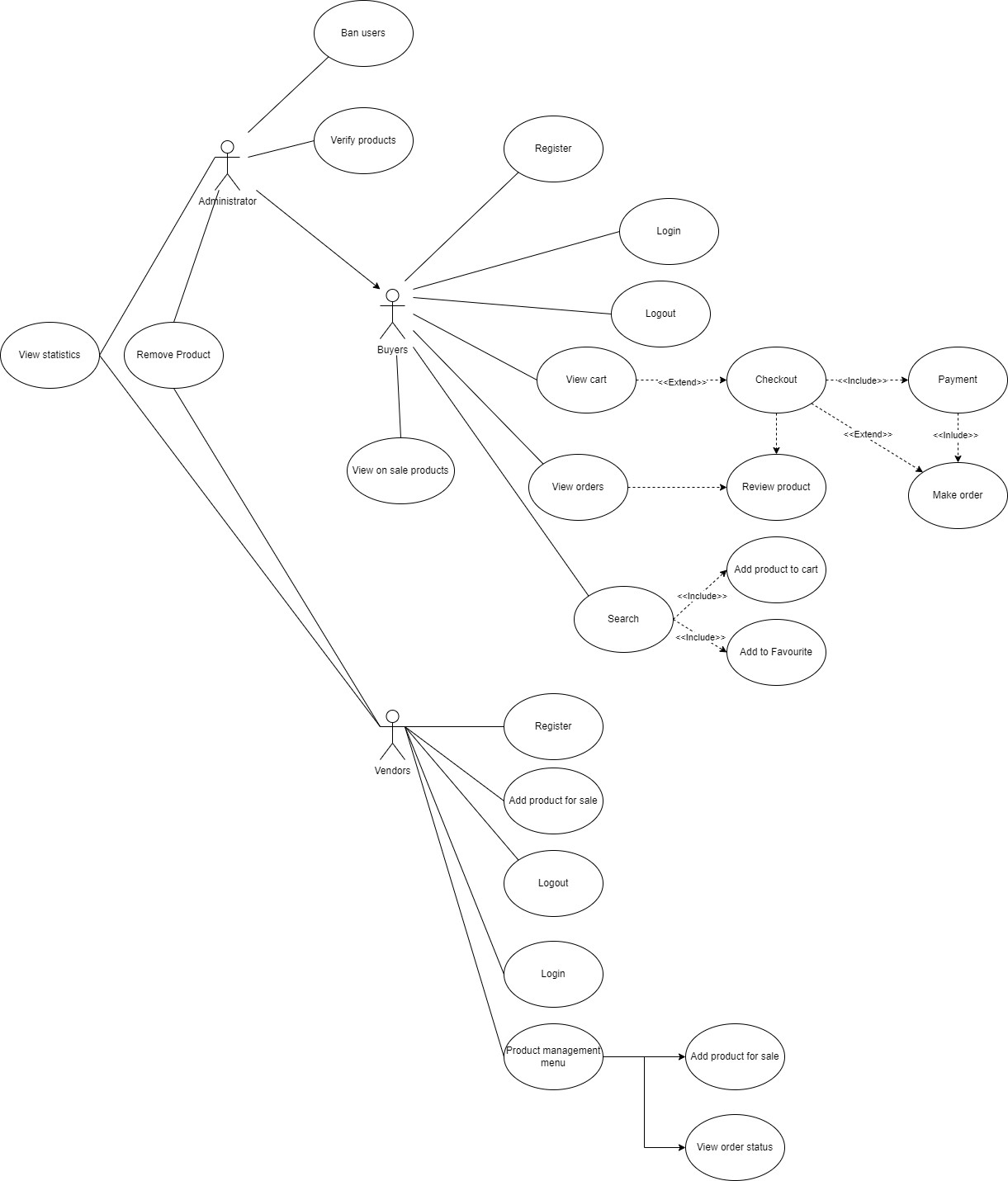
Database: Plannetscale

Diagram: DRAW.IO

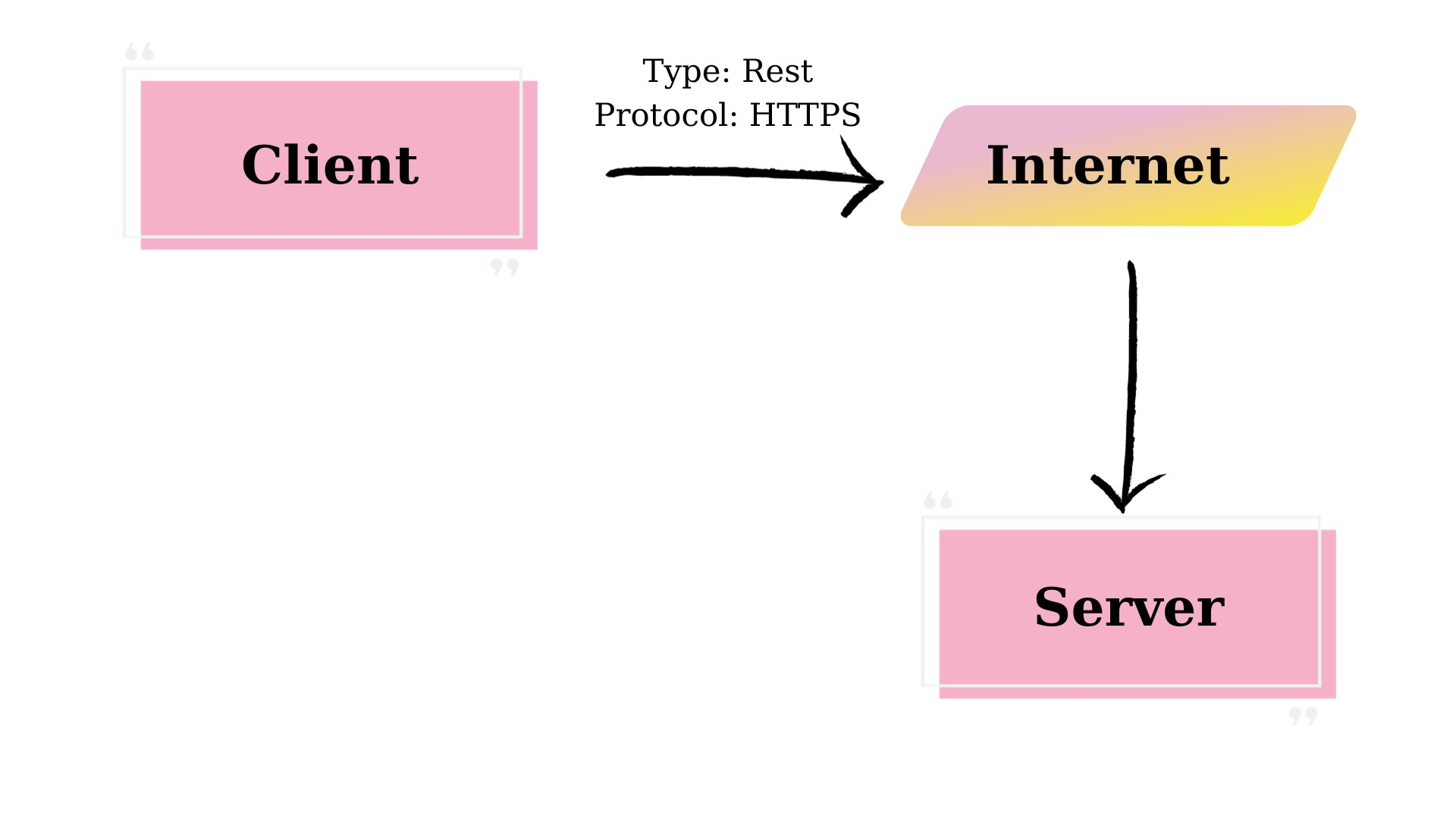
Database connection: Prisma

Schedule: Jira

# Use-Case Model

**

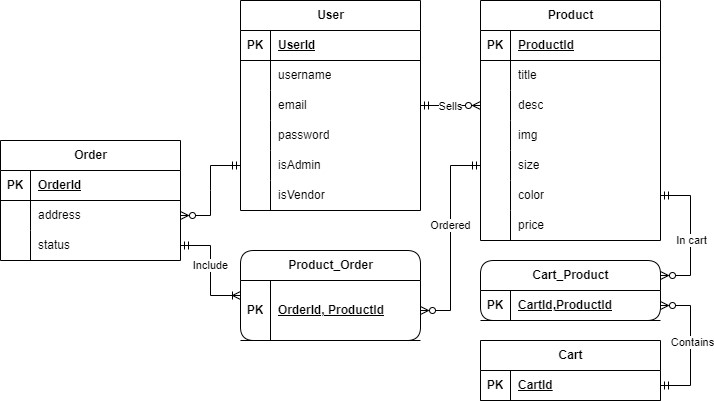
# Logical View

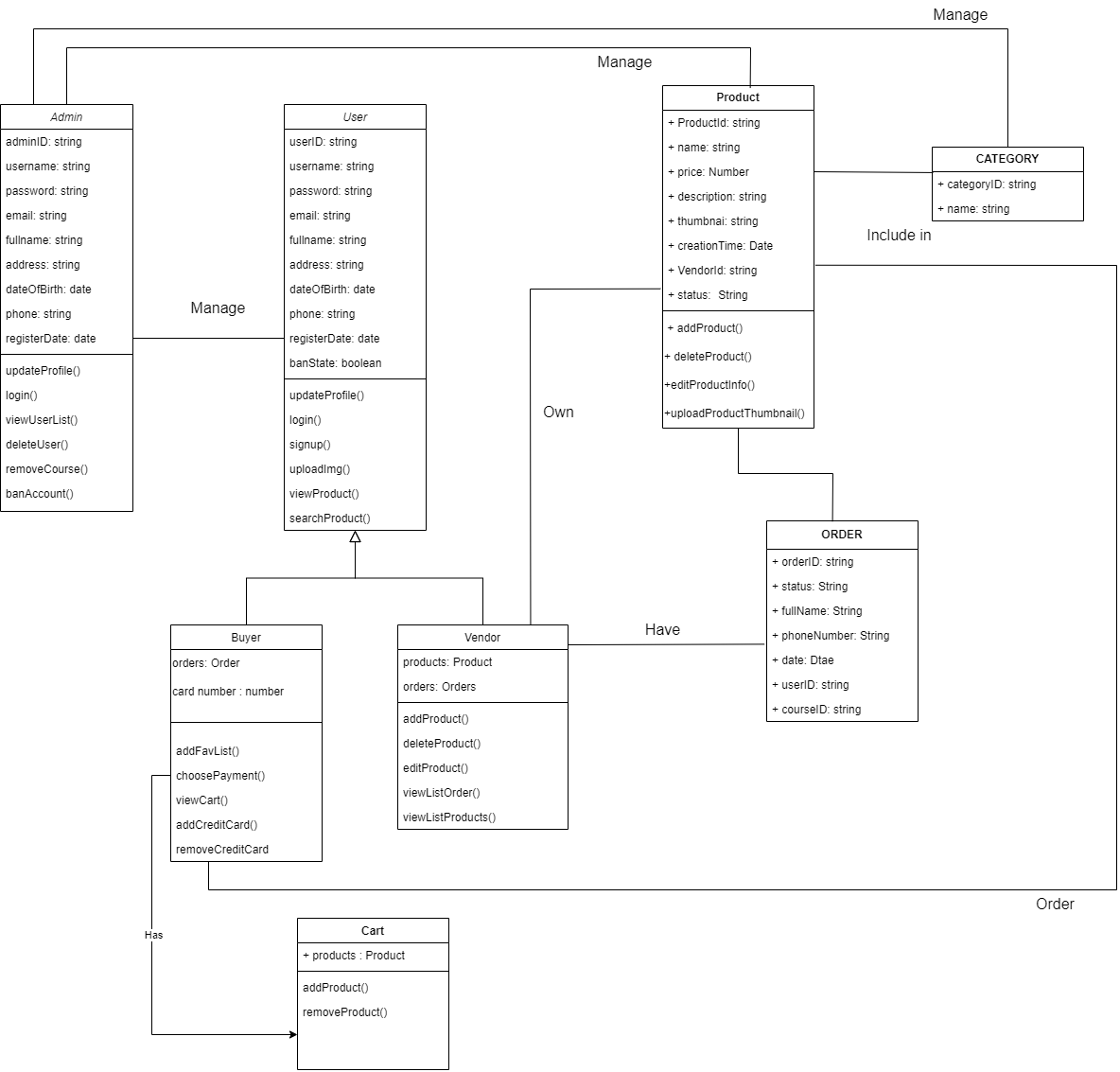


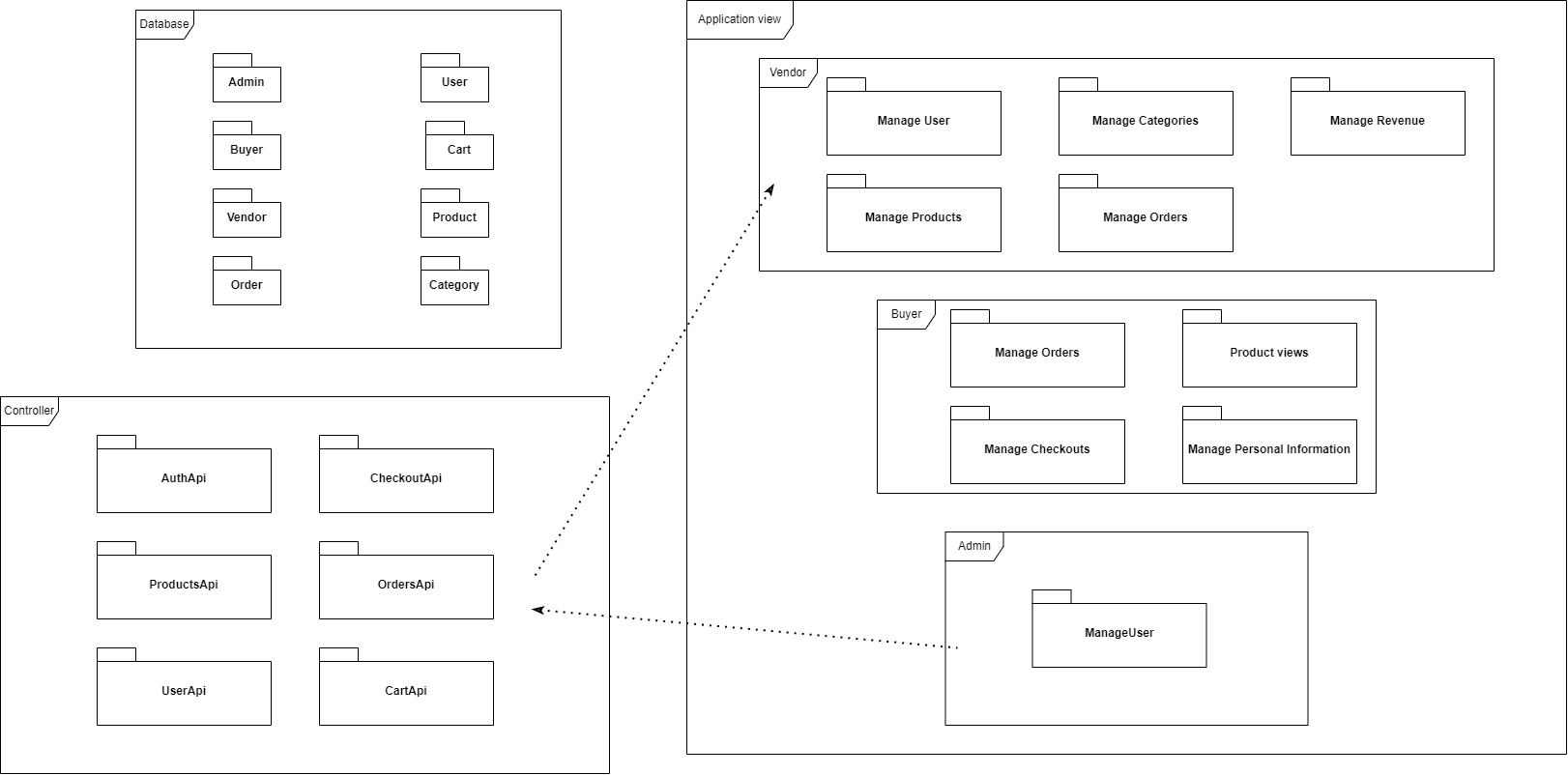
| | **Components** | | --- | | | **Responsibility** | | --- | |
| --- | --- | --- | --- |
| | Client | | --- | | | ● Present users with an HTML-based  user interface accessible through a  web browser  ● Interact with Server component to  submit, retrieve and show data from  database, also create users and  identify users. | | --- | |
| | Server | | --- | | | ● Handle all requests from the client  ● Response clients with appropriate  data  ● Communicate with the database by  executing some query commands  according to the client’s request | | --- | |

## Component: Server

Language: Javascript, Typescript







# Deployment

This system is hosted on a distant server since it is a web application (Vercel). Another hosting space will house the database(Plannetscale).The backend does all of the work, saving the client machine from using a lot of CPU power. When creating statistical graphs, all data processing is done at the front end, requiring a certain degree of speed from the client computer.

# Implementation View