Premier University

# Department of Computer Science & Engineering



**Final Year Project Report**

**On**

**Vaccine Center**

# Submitted by

Antar Nandi

ID: 1603110201203

***In partial fulfillment of the degree of***

Bachelor of Science in Computer Science & Engineering

# Under the Guidance of

Mr. Anik Sen

Assistant Professor

Department of Computer Science & Engineering

Premier University, Chittagong

May 2022

**Declaration**

In partial fulfillment of the requirements for the degree of Bachelor of Science, the project report is submitted to Department of Computer Science and Engineering, Premier University. I hereby declare that the work provided in this paper was completed under the supervision of Mr. Anik Sen, Assistant Professor, Department of Computer Science & Engineering, Premier University and that materials from other researchers' work are mentioned as references. This project report has never been submitted before, in whole or in part, for any degree.

…………………………

Antar Nandi

ID: 1603110201203

## 

## CERTIFICATE OF APPROVAL

This project title “Vaccine Center” submitted by Antar Nandi (ID: 1603110201203) has been accepted as satisfactory in fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering (CSE) as B.Sc. Engineering to be awarded by Premier University, Chittagong.

**………………………….**

## Prof. Dr. Taufique Sayeed

Chairman

Department of Computer Science and Engineering

Premier University Chittagong

**…………………………**

## Mr. Anik Sen

Assistant Professor

Department of Computer Science and Engineering

Premier University Chittagong

**ACKNOWLEDGEMENT**

First and foremost, when presenting this project, i express my heartfelt gratitude to Almighty for his grace and blessings, which enabled me to successfully finish this project work.

I'd like to take this occasion to thank my esteemed supervisor, Mr. Anik Sen Sir, Assistant Professor, Department of Computer Science & Engineering, Premier University, Chittagong. This project executed successfully under his supervision. His continual assistance and desire to help grasp this project and its manifestation in great depth of knowledge aided me in completing the assignment. Throughout the advancement of my work, his unwavering support, cooperation, inspiration, amazing direction, constructive criticism, and fantastic suggestions helped me make this work considerably better. I am cordially grateful to him and i really feel blessed to have him as my supervisor.

I am also grateful to Professor Dr. Taufique Sayeed, Chairman, Department of Computer Science and Engineering, Premier University for his unwavering encouragement and support. Finally, i'd want to express my gratitude to my parents, whose blessings and appreciations aided me in finally finishing this project. I'd also like to convey my gratitude and respect to my parents for their unwavering support, without which this work would not have been possible.

Last but not least, i'd like to express my gratitude to my parents. I appreciate my family's love, as well as their moral and financial support. I owe a debt of gratitude to my friends, classmates, seniors, juniors, and lab assistants, who have provided me with unwavering support and assistance in every way. I am extremely fortunate to have such a kind person as them by my side.

**ABSTRACT**

“Vaccine Center” is a system that helps users to get vaccine. Admin of this system can add vendor and purchase vaccines from them. Admin can add patient with specific vaccine and patient’s information. He/she can generate report for patients. Automated reminders are sent by “Vaccine Center” so that patients can update about their vaccination date. Filtering by different parameters might also help admin quickly find the information admin looking for. Admin also can generate report for purchase, sales and stock. Therefore, i have built an interactive and effective website called “Vaccine Center”.

## 

**Table of Contents**

## Chapter 1 1

## Introduction 1

* 1. Introduction 1
  2. Objectives 1
  3. Motivation 1
  4. Summary 2

## Chapter 2 3

## Literature Review 3

* 1. Introduction 3
  2. Review of Existing Systems 3
  3. Comparison of Our System 3
  4. Necessity of Methodology 4
  5. Software Development Life Cycle (SDLC) 4
     1. Feasibility Study 5
     2. Requirement Gathering and Analysis 5
     3. System Specification 5
     4. System Design 5
     5. Program Design and Coding 5
     6. Testing 5
  6. Software Process Model 6
  7. Summary 6

## Chapter 3 7

## Software Requirement Specification 7

* 1. Introduction 7
  2. Requirement Engineering 7
     1. Requirement Analysis 7
     2. Requirement Specification 9
     3. Requirement Validation 10

|  |  |  |
| --- | --- | --- |
| 3.3 Specific Requirements |  | 11 |
| 3.3.1 User Interface |  | 11 |
| 3.3.2 Hardware Interface |  | 11 |
| 3.3.3 Software Interface |  | 11 |
| 3.4 Software Requirement Specification |  | 12 |
| 3.4.1 User Requirements |  | 12 |
| 3.4.2 System Requirements |  | 12 |
| 3.4.3 Software Requirements |  | 12 |
| 3.5 Summary |  | 12 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Chapter 4 13** | | |  |
|  | **Normalization and Data Dictionary 13** | | |  |
|  | 4.1 Introduction 13 | | |  |
|  | 4.2 Database 13 | | |  |
|  | 4.3 Database Management System (DBMS) 13 | | |  |
|  | 4.4 Database Normalization 13 | | |  |
|  | 4.5 Normalization Rule 13 | | |  |
|  | 4.5.1 First Normal Form (1NF) 14 | | |  |
|  | 4.5.2 Second Normal Form (2NF) 14 | | |  |
|  | 4.5.3 Third Normal Form (3NF) 14 | | |  |
|  | 4.5.4 Boyce-Codd Normal Form (BCNF) 15 | | |  |
|  | 4.5.5 Fourth Normal Form (4NF) 15 | | |  |
|  | 4.6 Data Dictionary 15 | | |  |
|  | 4.7 Summary 26 | | |  |
|  |  | | |  |  |
|  | **Chapter 5 27** | | |  |
|  | **Modeling 27** | | |  |
|  | 5.1 Introduction 27 | | |  |
|  | 5.2 Use Case Diagram 27 | | |  |
|  | 5.3 Activity Diagram 29 | | |  |
|  | 5.3.1 Activity Diagrams for User 29 | | |  |
|  | 5.3.2 Activity Diagrams of Admin 32 | | |  |
|  | 5.4 ER Diagram 36  5.5 Summary 36  **Chapter 6 37**  **Software Implementation 37**  6.1 Introduction 37  6.2 Software Implementation 37  6.3 Web Server 39  6.4 Text Editor 40  6.5 Software 40  6.6 Snapshots 41  6.6.1 Product Add 41  6.6.2 Add New Deal Type 41  6.6.3 Subscription Add 42  6.6.4 Add Subscription by User 43  6.6.5 Deal Platform 44  6.6.6 Deal Type 45 | | |  |
|  | 6.6.7 Product Category 45  6.6.8 Add New Product Founder 46  6.6.9 User Register 47  6.6.10 User Log In 47  6.6.11 User Dashboard 48  6.6.12 All Subscription In User Dashboard 49  6.6.13 Membership Level 49  6.6.14 User Subscription Add 50  6.6.15 User Product Request 51  6.6.16 User Dashboard After Purchase 51  6.6.17 Purchase Subscription Information 52  6.6.18 Subscription Alert 53  6.6.19 Subscription Bulk Add 54  6.7 Summary 55  **Chapter 7 56**  **Discussion & Conclusion 56**  7.1 Introduction 56  7.2 Discussion 56  7.3 Limitation of The System 57  7.4 Future Work 57  7.5 Summary 57  **Reference 58** | | |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  |  | | |  |  |
|  |  |  |
|  |  |  |

**Table of Figures**

Figure 1: Software Development Life Cycle 4

Figure 2: Requirement Analysis Process Model 8

Figure 3: Requirement Specification Modeling 9

Figure 4: Non-functional Requirement 10

Figure 5: Use Case Diagram 28

Figure 7: Activity Diagram of Profile 29

Figure 8: Activity Diagram of Subscriptions 30

Figure 9: Activity Diagram of Accomplishment 30

Figure 10: Activity Diagram of Search 31

Figure 11: Activity Diagram of Request 31

Figure 12: Activity Diagram of All Users 32

Figure 13: Activity Diagram of Membership Request for Admin 33

Figure 14: Activity Diagram of Admin Product View 33

Figure 15: Activity Diagram of All Categories 34

Figure 16: Activity Diagram of Membership Request for User 35

Figure 17: Activity Diagram of Subscription for Admin And User 35

Figure 18 : ERD 36

# Introduction

**Chapter 1 Introduction**

A vaccine is a suspension of weakened, killed, or fragmented microorganisms or toxins or other biological preparation, such as those consisting of [antibodies](https://www.britannica.com/science/antibody), [lymphocytes](https://www.britannica.com/352799), or [mRNA](https://www.britannica.com/science/messenger-RNA), that is primarily used for to prevent diseases. A Vaccine Center provides complete, up-to-date and reliable information about vaccines to the general people. It helps us to find updated vaccines with necessary information.  The project “Vaccine Center” is a system that helps users to get vaccine. It helps Admin of this system to keep all the information about vaccines and patients. Admin of this system can add vendor and purchases vaccine from them. Users can get vaccine from this system and automated reminders are sent by “Vaccine Center” so that users can get update about their upcoming vaccination date. Admin of this system keep track of stored and sold vaccines. He can also see total sold vaccines and revenue from this system. Admin can generate report of stock, sales, purchase, patients list, vaccines list and pharmaceutical list using this system.

**Objectives**

* To add pharmaceuticals.
* To add vaccines and purchase from pharmaceuticals.
* To add one or multiple vaccines for a patient and patient information.
* To see a list of upcoming patients vaccination, vaccinated patients.
* To generate report for each feature.
* To manage report for sales, stock and purchase.
* To send notification for upcoming vaccination.

# Motivation

There are many vaccination center and they maintain their vaccines information and other details using hand written documents. But the main motive is to build this application “Vaccine Center” which will give the facilities to keep track of all information of vaccines and patients through database system.

# Summary

“Vaccine Center” is a system from where anyone can take vaccines and get notification for upcoming vaccination date. Admin of this system can also keep all information of users. Admin also can manage stock of vaccines. He can see also statistics of profit and loss of this system. He can filter data of this system and can generate report for each panel.

# Chapter 2 LITERATURE REVIEW

* 1. **Introduction**

A literature review is an overview of the previously published works on a specific topic. The term can refer to a full scholarly paper or a section of a scholarly work such as a book, or an article. The fundamental goal of writing a literature review is to convey to the reader the type of knowledge and thoughts that have been established on a topic, as well as the strengths and flaws.

# Review of existing systems

There are some Vaccination centers that are integrated with some medical system. These are –

# Evercare Hospital Limited

Evercare hospital vaccination system help user to get necessary information about different kind of vaccines stop paying for things you no longer need. Doses and price of those vaccines are given so that an user can get details about the vaccines. [1]

# United Hospital Limited

The Vaccination Center of United Hospital is providing its keen continues services. This department serves the indoor neonates, children & Adult by providing life savings vaccines. This department plays a vital role to improve the health status of community. This department maintains the excellent cold Chain for Vaccine Storage. [2]

# Comparison of our system

The main objective of this project is to use this system only for vaccination purpose. Admin of this system can keep detailed information of vaccines and patients. Admin can manage the stocks of vaccines.

# Necessity of Methodology

The ultimate objective of software engineering is to produce good quality maintainable software within reasonable time frame and at an affordable cost. This is achievable only if we have matured processes to produce it. For a mature process, it should be possible to determine in advance how much time and effort would be required to produce the final product.

Methodologies aim to place a degree of control over a software project, enabling it to be steered toward a successful conclusion through a proven series of steps and actions. It also allows for rapid delivery of high-quality software, and is a business approach that aligns development with both customer needs and company goals.

# Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aim to produces a high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. .



## Figure 2.1. Software development life cycle.

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

* + 1. **Feasibility Study**

A feasibility analysis or feasibility report is a way to evaluate whether or not a software project plan could be successful.

# Requirement Gathering and analysis

The requirement collecting and analysis phase of the SDLC is the most significant since it is here that the project team begins to comprehend what the customer wants from the project. The project team meets with the customer during the requirements gathering sessions to go over each requirement in depth.

# System specification

A System Needs Specification (SRD) is a process of Gather, analyze, and validate the information. It define the requirements and prototypes for new system. System specification evaluates the alternatives and prioritize the requirements. It examine the information needs of end-user and enhances the system goal. Software Requirement Specification (SRS) document, which specifies the software, hardware, functional, and network requirements of the system is prepared at the end of this phase.

# System Design

System Design Includes the design of application, network, databases, user interfaces, and system interfaces. It Transforms the SRS document into logical structure, which contains detailed and complete set of specifications that can be implemented in a programming language. It reviews the proposed design. Ensure that the final design must meet the requirements stated in SRS document.

# Program design and coding

Once the system design phase is over, the next phase is coding. In this phase, developers start building the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

# Testing

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement. During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fix the bugs and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system. There are two ways to implement my proposed system:

* + - * White Box Testing
      * Black Box Testing

# Software Process Model

A graphical representation of an object is provided by a software process model. It depicts a software system's activity. We followed the Agile model while designing the program.

Following are the Agile manifesto principles -

* **Individuals and interactions:** I have worked in my project individually. Individuals and interactions are valued in the Agile methodology.
* **Working software:** I built my method step by step to understand public demand and tested it with our supervisor. Because a working demo is the most effective approach to develop.
* **Customer collaboration:** My project is totally focused on my clients' requirements. Without consumer interaction, I won't be able to meet public demand and necessity. As a result, maintaining a steady client connection is the most important thing.
* **Responding to change:** Using the agile paradigm, i can create them quickly and continue to develop this system.

**Advantages of Agile Model:**

* Customer satisfaction is achieved by the constant distribution of software.
* Customer, testers, and developers communicate on a regular basis.
* During the development phase, requirements can be updated.

# Summary

The existing systems and the Software Development Life Cycle are discussed in this chapter (SDLC).

**Chapter 3**

**SOFTWARE REQUIREMENT SPECIFICATION**

* 1. **Introduction**

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

# Requirement Engineering

**Requirements engineering** refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system.

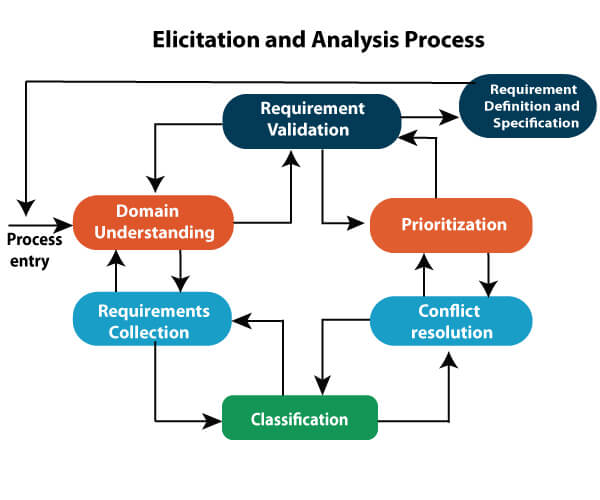
Requirement Engineering process consists of the following main activities:

* + - Requirement Analysis
    - Requirement Specification
    - Requirement Validation

# Requirement Analysis

Requirement analysis is significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements. This activity reviews all requirements and may provide a graphical view of the entire system. After the completion of the analysis, it is expected that the understandability of the project may improve significantly.

We can represent requirement analysis process as the below as shown in Figure 3.1:



## Figure 3.1. Requirement analysis process model.

* **Domain Understanding**

Learning about the industry in which software systems will be employed is the main focus of this level.

# Requirement Collection

In this section, the idea of the problem domain and the user's needs are discussed. The general public is the target market for this campaign.

# Conflict Resolution

The requirements will conflict if a system has many subscribers. Conflicts within our system are something we are very aware of. Our system is developed enough to deal with any potential conflicts.

# Prioritization

Prioritization, which involves identifying the most crucial requirements, among other things, requires interaction with the user..

# Requirement Validation

This phase determines whether they are complete and meet the subscriber's specific needs.

# Classification

This method organizes a disorganized collection of needs into multiple clusters. In our system, we have four types of user information that are stored in coherent clusters.

# Requirement Specification

The production of the requirements stage of the software development process is **Software Requirements Specifications (SRS)**(also called a**requirements document)**. This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system.

**Following are the features of a good SRS document:**

* **Correctness**

User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.

* **Completeness**

The SRS is complete if, and only if, it includes the following elements:

1. All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.
2. Definition of their responses of the software to all realizable classes of input data in all available categories of situations.
3. Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.

* **Consistency**

The SRS is consistent if, and only if, no subset of individual requirements described in its conflict.

* **Unambiguousness**

SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.

**Non Functional Requirement**

Non-functional requirements in an SRS document (software requirements specification) specify the quality attributes of a software system, such as responsiveness, usability, and reliability. As functional requirements indicate what a system must do, non-functional requirements support them and determine how the system must perform.

##### Different types of non-functional requirements:

* **Performance**

System performance is the most important quality in non-functional requirements and affects almost all the other preceding ones. Like the website’s load time should not be more than one second for users.

* **Reliability**

Reliability is the probability and percentage of the software performing without failure for a specific number of uses or amount of time.

* **Availability**

This feature defines the amount of time the system is running, the time it takes to repair a fault, and the time between lapses. Employers can post jobs on the website throughout the week at any time during the day.

* **Maintainability**

This feature indicates the average time and ease and rapidity with which a system can be restored after a failure. If the automated email services become unavailable, they can be under maintenance for approximately three hours.

* **Recoverability**

Recoverability is the ability to recover from a crash or a failure in the system and returning to full operations. If a major incident happens on the website and, the business must take measures to go back to being fully operational within short periods.

* **Security**

Security measures ensure your software’s safety against espionage. you don’t want anyone to have access to your sensitive data. Suppose only the users with the role “site admin” can view the applicant’s verified phone number.

* **Data integrity**

Data integrity refers to maintaining and assuring data accuracy and consistency over its entire lifecycle. If this factor is corrupted, data is lost due to a database error. The system shall maintain data integrity by keeping backups of all updates to the database for every record transaction.

* **Usability**

This feature concerns the users; it indicates how effectively they can learn and use a system. The website’s interface has to be user-friendly and easy to use.

* + 1. **Requirement Validation**

**Requirements validation** is the process of checking that requirements defined for development, define the system that the customer really wants. To check issues related to requirements, we perform requirements validation. We usually use requirements validation to check error at the initial phase of development as the error may increase excessive rework when detected later in the development process.

Different types of checks are performed on the requirements:

1. **Validity Check:** The functions proposed by stakeholders should be aligned with what the system needs to perform. You may find later that there are additional or different functions are required instead.
2. **Consistency Check:** Requirements in the document shouldn’t conflict or different descriptions of the same function
3. **Completeness Check:** The document should include all the requirements and constraints.
4. **Realism Check:** Ensure the requirements can actually be implemented using the knowledge of existing technology, the budget, schedule, etc.
5. **Verifiability:** Requirements should be written so that they can be tested. This means you should be able to write a set of tests that demonstrate that the system meets the specified requirements.

# Specific Requirements

**3.3.1 User Interface**

Because of the system's step-by-step processing, a user-friendly interface was required.

# 3.3.2 Hardware Interface

## Table 3.1. Hardware requirement table.

|  |  |
| --- | --- |
| Operating System | Windows 10, Linux, Unix etc. |
| Computer/Processor | Intel CORE i5 |
| Memory | At least 512 MB RAM |
| Web Server | PHP compatible web server |
| Required memory on the server | 2 GB |

**3.3.3 Software Interface**

**Table 3.2. Platform and tools table.**

|  |  |
| --- | --- |
| Database | MySQL |
| Web browser | Any PHP supported web browser such as Mozilla Firefox, Google Chrome, etc. |
| Editor | Visual Studio Code |
| Programming Language & Technologies | HTML, CSS, Bootstrap, JavaScript, Laravel. |

**3.4 Software Requirement Specification**

**3.4.1 User Requirements**

The user requirement(s) document (URD) or user requirement(s) specification (URS) is a document usually used in software engineering that specifies what the user expects the software to be able to do.

# 3.4.2 System Requirements

# System requirements are derived from stakeholder requirements and describe what the system must do based on the requiresments. System requirements describe specific functions necessary within a system to "satisfy" each stakeholder requirement.

# 3.4.3 Software Requirements

# The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client’s point of view.

# 3.5 Summary

Software requirement specifications are discussed in this chapter. What software and hardware interfaces are needed to develop this platform, as we mentioned in this chapter.

# Chapter 4 Normalization and Data Dictionary

* 1. **Introduction**

We'll discuss about databases, database management systems (DBMS), database normalization, and data dictionaries in this chapter.

# Database

# Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc. In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable. A **database** is an organized collection of data, so that it can be easily accessed and managed. The **mainpurpose** of the database is to operate a large amount of information by storing, retrieving, and managing data. There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server. Modern databases are managed by the database management system (DBMS).

.

# Database Management System (DBMS)

A database management system (DBMS) is software that allows you to create, define, and manipulate databases. It enables the user to conveniently store, process, and analyze data. A database management system (DBMS) provides us with an interface or tool that allows us to execute various tasks such as building a database, putting data in it, updating data, adding tables in the database, and so on. Some DBMS examples include MySQL, Microsoft Access, SQL Server, FileMaker, Oracle, RDBMS, Clipper, and FoxPro.

# Database Normalization

Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. In most cases, normalization means separating existing tables into several ones.

# Normalization Rule

At various phases, normalization rules are used to edit or update bibliographic metadata.

Normalization rules are divided into the following normal forms:

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce Codd Normal Form (BCNF)
5. Fourth Normal Form (4NF)

# First Normal Form (1NF)

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relation is in first normal form if it does not contain any **composite**or**multi-valued attribute.** A relation is in first normal form if every attribute in that relation is singled valued attribute.

The first normal form states that:

* There are only Single Valued Attributes.
* Attribute Domain does not change.
* There is a unique name for every Attribute/Column.
* The order in which data is stored does not matter.

# Second Normal Form (2NF)

The second normal form (2NF) is a database normalization normal form. If a relation meets both of the following conditions, it is said to be in the "second normal form."

* + - * It should be in its original form (First normal Form).
      * It has no non-prime attributes that are functionally dependent on any suitable subset of any relation's candidate key.

# Third Normal Form (3NF)

The third normal form (3NF) is a database normalization normal form. If a relation meets both of the following conditions, it is said to be in the "third normal form."

* A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
* 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
* If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

# Boyce-Codd Normal Form (BCNF)

The Boyce-Codd Normal Form, also known as 3.5 Normal Form, is an extension of the third normal form. The following two conditions must be met for a table to satisfy the Boyce-Codd Normal Form:

* + - 1. It needs to be written in the Third Normal Form
      2. A should be a super key for any dependency A → B

# Fourth Normal Form (4NF)

Fourth normal form (4NF) is a database normalization normal form. If a relation meets both of the following conditions, it is said to be in the "fourth normal form."

* A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
* For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

# Data Dictionary

A data dictionary is a well-structured document in which all data pieces are applicable to the system and have accurate descriptions, allowing users and system analysts to grasp inputs, outputs, and even intermediate calculations with ease.

In users table id is the primary key. Other fields are name, email, password, created\_at, updated\_at. This table is used to store data of Admins information and credentials on our platform.

**Table 4.1 admin**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| id | bigint(20) | Primary Key | No | None |
| name | varchar(255) | - | No | None |
| email | varchar(255) | - | Yes | None |
| password | varchar(255) | - | Yes | None |
| created\_at | timestamp | - | No | NULL |
| updated\_at | timestamp | - | Yes | NULL |

In pharmaceuticals table id is the primary key. Other fields are p\_name, address, status, created\_at, updated\_at. This table is used to store pharmaceuticals information on our platform.

.

**Table 4.2 pharmaceuticals**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| id | bigint(20) | Primary Key | No | None |
| p\_name | varchar(255) | - | No | None |
| address | varchar(255) | - | Yes | Null |
| status | varchar(255) | - | Yes | None |
| created\_at | timestamp | - | Yes | NULL |
| updated\_at | timestamp | - | Yes | NULL |

In vaccines table id is the primary key. Other fields are vaccine\_name, doses, status, is\_continuous, created\_at, updated\_at. This table is used to store vaccines information on our platform.

**Table 4.3 vaccines**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| product\_id | bigint(20) | Foreign Key | No | None |
| code | varchar(255) | - | Yes | NULL |
| features | varchar(255) | - | No | None |
| plan\_name | varchar(255) | - | No | None |
| name | varchar(255) | - | Yes | NULL |
| deal\_price | varchar(255) | - | Yes | NULL |
| screenshot | varchar(255) | - | Yes | NULL |
| link | varchar(255) | - | Yes | NULL |
| regular\_price | varchar(255) | - | Yes | NULL |
| created\_at | timestamp | - | No | Current\_timestamp() |
| updated\_at | timestamp | - | No | NULL |

In purchases table id is the primary key. vaccine\_id and pharma\_id is foreign key .Other fields are mfg\_date, exp\_date, qnty, pur\_date, price, v\_price, discount, created\_at, updated\_at. This table is used to store vaccines information on our platform.

.

**Table 4.4 purchases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint (20) | Primary Key | No | None |
| product\_founder\_id | int (11) | Foreign Key | No | None |
| Product\_caregory\_id | varchar (255) | Foreign Key | No | None |
| name | varchar (255) | - | No | NULL |

In purchases table id is the primary key. vaccine\_id and pharma\_id is foreign key .Other fields are mfg\_date, exp\_date, qnty, pur\_date, price, v\_price, discount, created\_at, updated\_at. This table is used to store vaccines information on our platform.

**Table 4.5. wp\_product\_categories**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| name | varchar (255) | - | No | None |
| slug | varchar (255) | - | No | None |
| description | varchar (255) | - | Yes | NULL |
| created\_at | timestamp | - | Yes | current\_timestamp() |
| updated\_at | timestamp | - | Yes | NULL |

In wp\_product\_founders table ID is the primary key. Other fields are avatar, twitter, facebook, website, email, youtube, linkedin, country, bio, created\_at and updated\_at. Founders info are stored in this table.

**Table 4.6.wp\_product\_founders.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| name | varchar (255) | - | Yes | None |
| avatar | varchar (255) | - | Yes | NULL |
| twitter | varchar (255) | - | Yes | NULL |
| facebook | varchar (255) | - | Yes | NULL |
| website | varchar (255) | - | Yes | NULL |
| email | varchar (255) | - | Yes | NULL |
| youtube | varchar (255) | - | Yes | NULL |
| linkedin | varchar (255) | - | Yes | NULL |
| country | varchar (255) | - | Yes | NULL |
| bio | varchar (255) | - | Yes | NULL |
| created\_at | timestamp | - | No | current\_timestamp() |
| updated\_at | timestamp | - | Yes | NULL |

In wp\_product\_rating table ID is the primary key and product\_id, user\_id are foreign keys. Other fields are review,rating, created\_at and udpated\_at. This table is used to store rating of a deal.

**Table 4.7. wp\_product\_raitng.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| product\_id | bigint(20) | Foreign Key | No | None |
| user\_id | bigint(20) | Foreign Key | No | None |
| rating | Float(5.2) | - | Yes | NULL |
| review | varchar (255) | - | Yes | NULL |
| created\_at | timestamp | - | No | current\_timestamp() |
| updated\_at | timestamp | - | No | NULL |

In wp\_product\_with\_deal\_platforms table, ID is the primary key and product\_id, deal\_platform\_id are the foreign key. Other fields are created\_at and updated\_at. This table is used to store multiple deals info on single platform.

**Table 4.8. wp\_product\_with\_deal\_platform.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| product\_id | bigint(20) | Foreign Key | No | None |
| deal\_platform\_id | bigint(20) | Foreign Key | No | None |
| created\_at | timestamp | - | Yes | NULL |
| updated\_at | timestamp | - | Yes | NULL |

In wp\_subs table, ID is the primary key and user\_id, product\_id, pricing\_plan\_id, subs\_tag\_id, deal\_platform\_id, deal\_type\_id,subs\_folder\_id, are foreign key . Other fields are migration, name, stacked, price, status, bought\_date, refund\_date, redeem\_date, payment\_processor and plan. This table is used to store the data of subscriptions, user has purchased from various deals.

**Table 4.9. wp\_subs.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| user\_id | bigint(20) | Foreign Key | No | None |
| product\_id | bigint(20) | Foreign Key | Yes | NULL |
| Pricing\_plan\_id | bigint(20) | Foreign Key | Yes | NULL |
| subs\_tag\_id | bigint(20) | Foreign Key | Yes | NULL |
| deal\_platform\_id | bigint(20) | Foreign Key | Yes | NULL |
| deal\_type\_id | bigint(20) | Foreign Key | Yes | NULL |
| subs\_folder\_id | bigint(20) | Foreign Key | Yes | NULL |
| name | varchar (255) | - | No | NULL |
| stacked | varchar (255) | - | Yes | NULL |
| price | varchar (255) | - | Yes | NULL |
| status | varchar (255) | - | Yes | NULL |
| bought\_date | date | - | Yes | NULL |
| refund\_date | date | - | Yes | NULL |
| redeem\_date | date | - | Yes | NULL |
| signup\_email | varchar (255) | - | Yes | NULL |
| payment\_processor | varchar (255) | - | Yes | NULL |
| login\_id | varchar (255) | - | Yes | NULL |
| currency | varchar (255) | - | Yes | NULL |
| invoice | varchar (255) | - | Yes | NULL |
| cycle | varchar (255) | - | Yes | NULL |
| folder | varchar (255) | - | Yes | NULL |
| retail\_price | varchar (255) | - | Yes | NULL |
| listing\_price | varchar (255) | - | Yes | NULL |
| sold\_for | varchar (255) | - | Yes | NULL |
| for\_sale | tinyint(1) | - | No | NULL |
| Notes | varchar (255) | - | Yes | NULL |
| privacy | varchar (255) | - | Yes | NULL |
| plan | varchar (255) | - | Yes | NULL |
| created\_at | timestamp | - | No | NULL |
| updated\_at | timestamp | - | Yes | NULL |

In wp\_subs\_folder table, ID is the primary key and user\_id is the foreign key. Other fields are name, slug, created\_at and updated\_at. This table is used to store folder data of subs.

**Table 4.10. wp\_subs\_folder.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| user\_id | bigint(20) | Foreign Key | No | None |
| name | varchar (255) | - | No | None |
| slug | varchar (255) | - | No | None |
| created\_at | timestamp | - | No | NULL |
| updated\_at | timestamp | - | No | NULL |

In wp\_subs\_reminder table, ID is the primary key and subs\_id is the foreign key. Other fields are one\_contact, one\_note, one\_date, two\_contact, two\_note, two\_date, three\_contact, three\_notes, three\_date created\_at, updated\_at. Here in this table we are storing reminder info of a subscription.

**Table 4.11. wp\_subs\_reminder.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| subs\_id | bigint(20) | Foreign Key | No | None |
| one\_contact | varchar (255) | - | Yes | NULL |
| one\_note | varchar (255) | - | Yes | NULL |
| one\_date | date | - | Yes | NULL |
| two\_contact | varchar (255) | - | Yes | NULL |
| two\_note | varchar (255) | - | Yes | NULL |
| two\_date | date | - | Yes | NULL |
| three\_contact | varchar (255) | - | Yes | NULL |
| three\_note | varchar (255) | - | Yes | NULL |
| three\_date | date | - | Yes | NULL |
| created\_at | timestamp | - | Yes | NULL |
| updated\_at | timestamp | - |  | NULL |

In wp\_subs\_tags table, ID is the primary key and user\_id is foreign key. Other fields are name, slug, created\_at and updated\_at. Subscription tag info is storing in this table.

**Table 4.12. wp\_subs\_tags.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Index** | **Null** | **Default** |
| ID | bigint(20) | Primary Key | No | None |
| user\_id | bigint(20) | Foreign Key | No | None |
| name | varchar (255) | - | No | None |
| slug | varchar (255) | - | Yes | NULL |
| created\_at | timestamp | - | Yes | NULL |
| updated\_at | timestamp | - | Yes | NULL |

**4.7 Summary**

We discussed about databases, database management systems (DBMS), normalization, and data dictionaries in this chapter.

# Introduction

**Chapter 5 Modeling**

We'll discuss about our project's use case diagram, activity diagram, and ER diagram in this chapter.

# Use Case Diagram

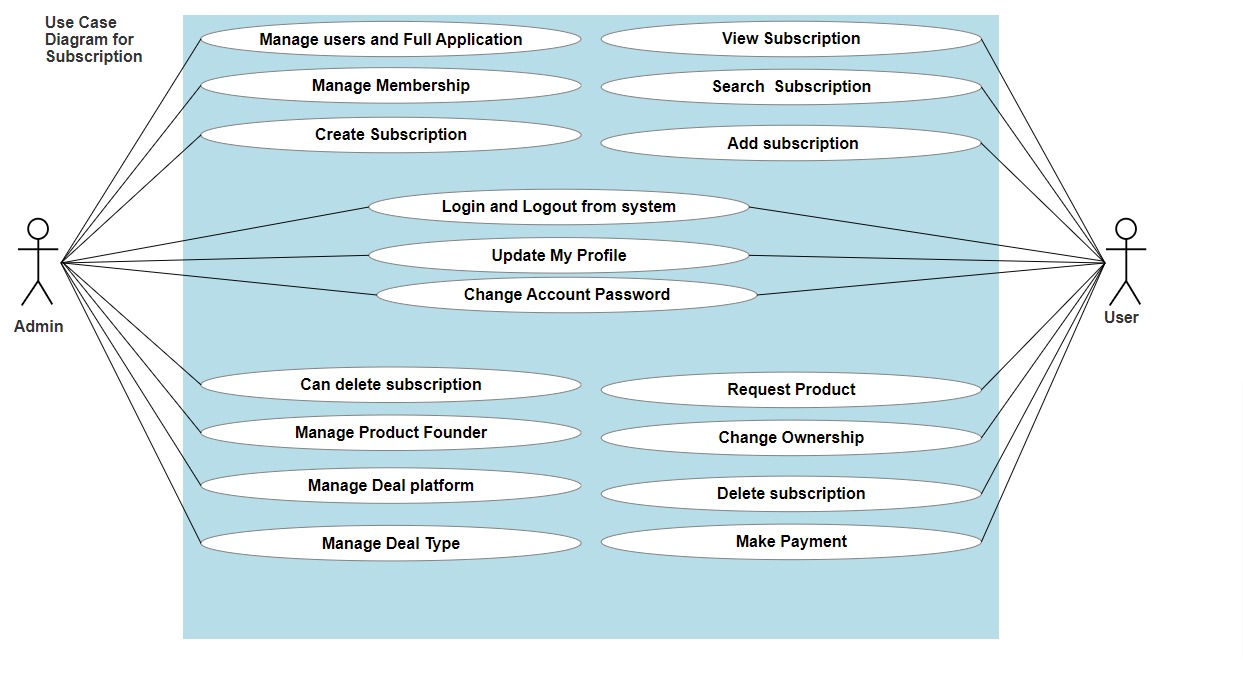
Use case diagrams are the scenario-based method, which identifies the actions of communication.

# Admin

* Admin can see the list of users of this platform. He can approve or disapprove user
* Admin can add and manipulate all the membership functionality. He can approve and disapprove membership update request
* Admin can see, add, update and delete products on this platform
* Admin can see, add, update and delete subscription on this platform
* Admin can see, add, update and delete subscription deal type on this platform
* Admin can view and update his profile
* Admin can see, add, update and delete founders on this platform
* Admin can see, add, update and delete rating
* Admin can see list of membership on this platform
* Admin can see users request of adding products
* Admin can track ownership change of subscription

**User**

* Users can register and login on this platform
* Users can view and update his profile
* Users can see, update and delete subscription from his profile
* User can buy subscription on this platform
* User can see list of subscription on this platform
* User can search subscription using this platform
* User can request to update the membership to premium after completing the subscription



**Figure 5.1. Use case diagram of user and admin.**

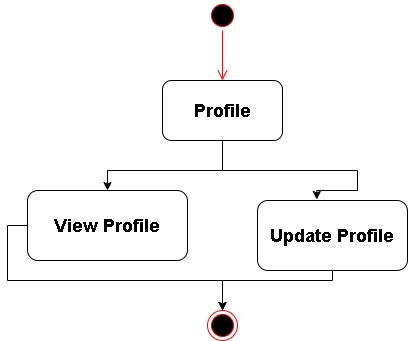
# 5.3 Activity Diagram

## 5.3.1 Activity Diagrams for User

On this platform, there are three kinds of users. They are user, Premium, & Admin. All of the users will get a separate profile section. In this profile section, they can see their subscription information, expire date and accomplishment information as well. They will also get search functionality.

**Profile**

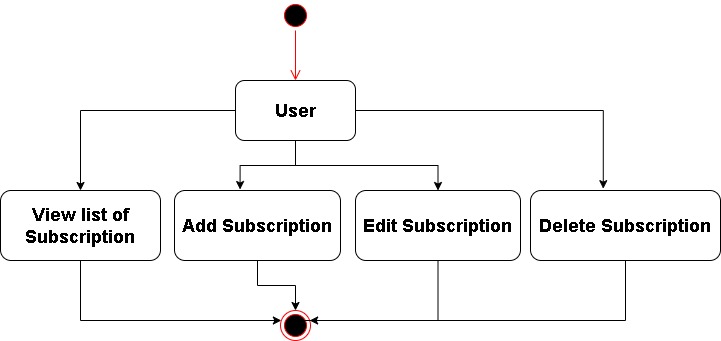
In this platform, every user will get an individual profile section. He can see his profile information and can update information as well like country, region, mobile number.



**Figure 5.2. Activity diagram of profile.**

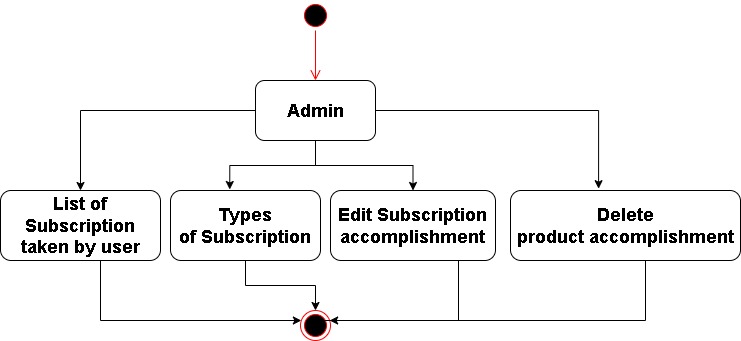
**Subscription**

Every user can see his subscription information in his profile. Users can add new subscription information. Users can update or delete their subscription information as well.



**Figure 5.3. Activity diagram of subscription.**

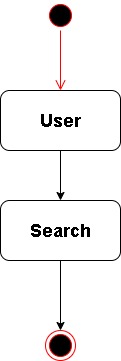
**Accomplishment**

Every user can see his accomplishment information in his profile. Users can add new subscription accomplishment information. User can update or delete his accomplishment information as well. 

**Figure 5.4. Activity diagram of accomplishment.**

**Search**

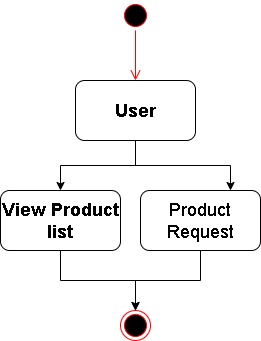
Every user can search for subscription. They can also filter the search result as well.



**Figure 5.5. Activity diagram of search.**

**User Product Request**

Every user can Request for subscription.

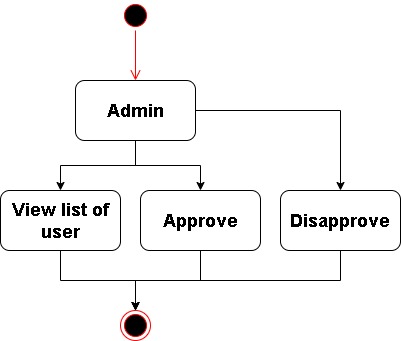


**Figure 5.6. Activity diagram of user product request.**

## 5.3.2 Activity Diagrams of Admin

**All Admin**

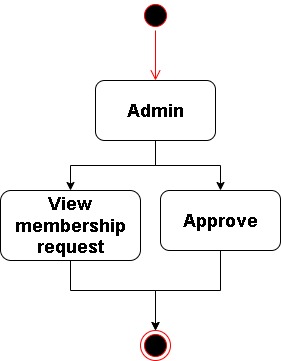
Admin can see the list of users in our platform. He can approve or disapprove users by checking their information.



**Figure 5.7. Activity diagram of all users.**

**Membership Request** **(For admin)**

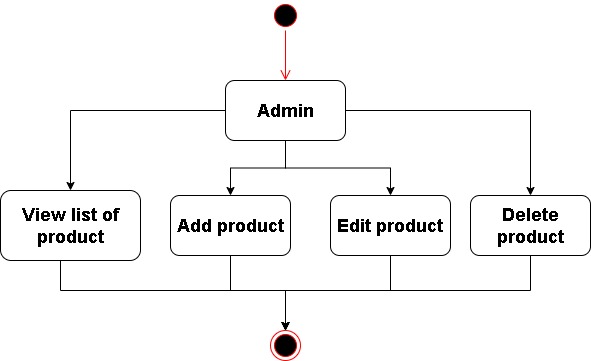
After taking subscription, users can update their membership to Premium by providing necessary information. Again, a premium can update his membership to platinum by providing necessary information. Admin can see all the requests of the users and after checking he can approve membership update requests.



**Figure 5.8. Activity diagram of membership request for Admin.**

**Admin Product View**

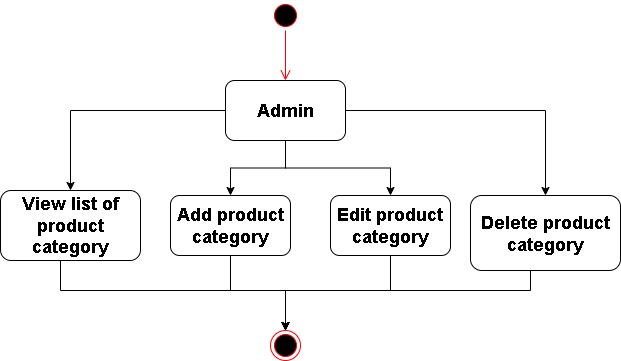
After Adding products, admin can see all the product list. Admin can edit and delete all the products.



**Figure 5.9. Activity diagram of Admin Product view.**

**Admin Product Category**

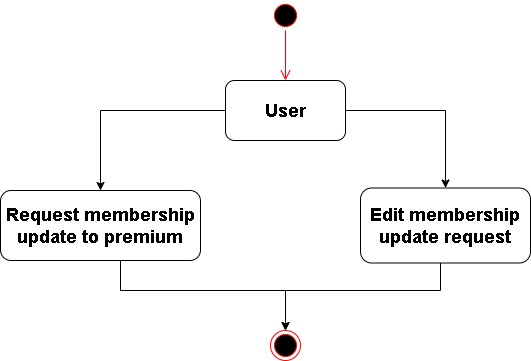
Admin can see the list of categories in our platform. He can edit or delete category.



**Figure 5.10. Activity diagram of all categories.**

**Membership Request (For User)**

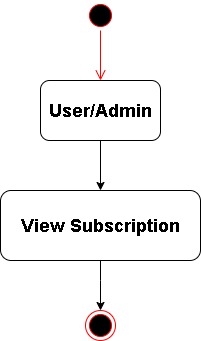
After completing subscription, user can request to update their membership to premium with necessary information. If they provide wrong information accidentally, they can update their request also.



**Figure 5.11. Activity diagram of membership request for user.**

**Admin / User**

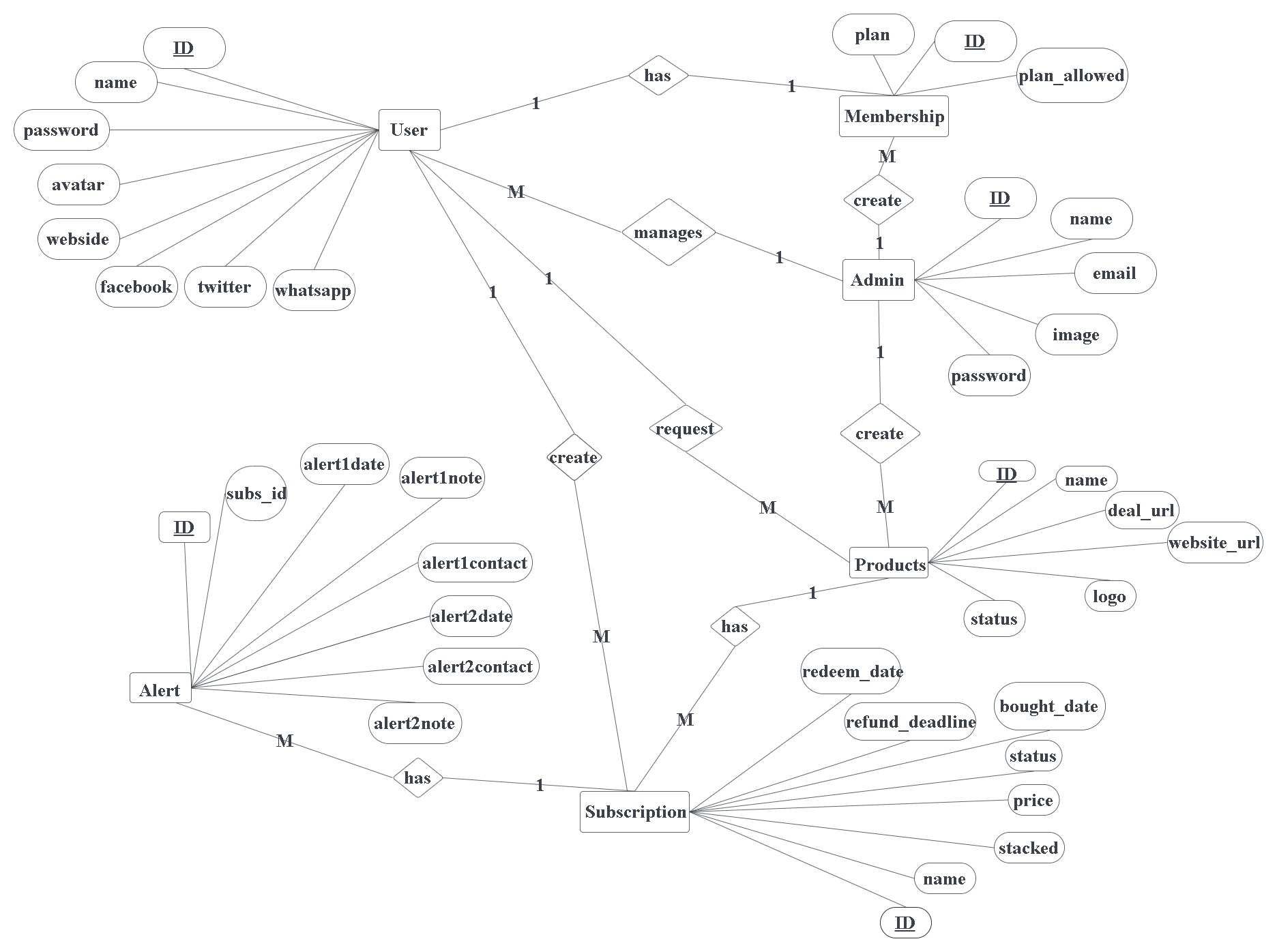
They can see all the subscriptions that are made by Admin on our platform.



**Figure 5.12: Activity diagram of subscription for admin and user**.

# 

# 5.4 ER Diagram



**Figure 5.13. ERD.**

# 5.5 Summary

We demonstrated how our product works by utilizing a use case diagram and an activity diagram.

# Introduction

**Chapter 6 SOFTWARE IMPLEMENTATION**

I have searched for all known and available materials that could assist us as we made the decision to conduct the project. Implementing a website is difficult since it entails storing and retrieving a large amount of data. We employed a relational data-based system and an object-oriented database connection to make string and searching more reliable, easier, and faster than ever before. PHP, Wordpress, HTML, CSS, JavaScript, Vue.js, Tailwind, and Headless UI have all made significant contributions to the development of websites. We've employed all of them to ensure the integrity of our project.

# Software Implementation

We implement our software using given below:

* + 1. Laravel
    2. HTML
    3. CSS
    4. JavaScript
    5. Bootstrap
    6. MySQL

**Laravel**

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable, creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching. Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality. Happy developers make the best code. To this end, we've attempted to combine the very best of what we have seen in other web frameworks, including frameworks implemented in other languages, such as Ruby on Rails, ASP.NET MVC, and Sinatra. Laravel is accessible, yet powerful, providing powerful tools needed for large, robust applications. A superb inversion of control container, expressive migration system, and tightly integrated unit for testing

# HTML

HTML stands for Hypertext Markup Language. It allows the user to create and structure sections, paragraphs, headings, links, and block quotes for web pages and applications.HTML is not a programming language, meaning it doesn’t have the ability to create dynamic functionality. Instead, it makes it possible to organize and format documents, similarly to Microsoft Word.HTML documents are files that end with a .html extension. You can view then using any web browser (such as Google Chrome, Safari, or Mozilla Firefox). The browser reads the HTML file and renders its content so that internet users can view it.

# CSS

CSS (Cascading Style Sheets) allows you to create great-looking web pages, but how does it work under the hood? This article explains what CSS is, with a simple syntax example, and also covers some key terms about the language. CSS can be used for very basic document text styling For example changing the color and size of headings and links. It can be used to create layout. Besides turning a single column of text into a layout with a main content area and a sidebar for related information. It can even be used for effects such as animation. Have a look at the links in this paragraph for specific examples.

# JavaScript

JavaScript is one of the most popular scripting languages. At present, millions of web pages improve their design, validate forms, detect browsers, create cookies and much more. It runs almost all the major browsers such as google chrome, internet explorer firefox, opera.

* It is a lightweight programming language
* JavaScript directly embedded to HTML pages

● Easy to use require no license [8]

# Vue.js

Vue (pronounced /vjuː/, like view) is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries. [9]

# MySQL

MySQL is a freely available open source RElational Database Management System (RDBMS) that uses Structured Query Language (SQL). SQL is the most popular language for adding, accessing and managing content in a database. It is most noted for its quick processing, proven reliability, ease and flexibility of use. [10]

# Tailwind

Tailwind CSS is basically a utility-first CSS framework for rapidly building custom user interfaces. It is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override. [11]

# Web Server

A web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computer’s HTTP clients. [12]

# Apache

Apache is the most widely used web server software which is developed and maintained by Apache Software Foundation. It can be customized to meet the needs of many different environments by using extensions and modules. Most of the WordPress hosting providers use Apache for their web server software. [13]

# Web Browser

A browser is a software application used to locate, retrieve and display content on the World Wide Web, including web pages, images, video and other files. We used Google Chrome to develop and run the system. Because it is more reliable for databases.[14]

# Text Editor Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor. It runs on desktop and is also available for Windows, macOS and Linus. That provides different facilities with build-in support for JavaScript, TypeScript and Node.js etc. [15]

* IntelliSense for programming language
* Efficient command palette
* Highly integrated version control
* Debugging
* Side by side edition on different files

# Software XAMPP

XAMPP is a free and open source, cross-platform where web server, solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming language. It is very easy to use because of its several functionality.

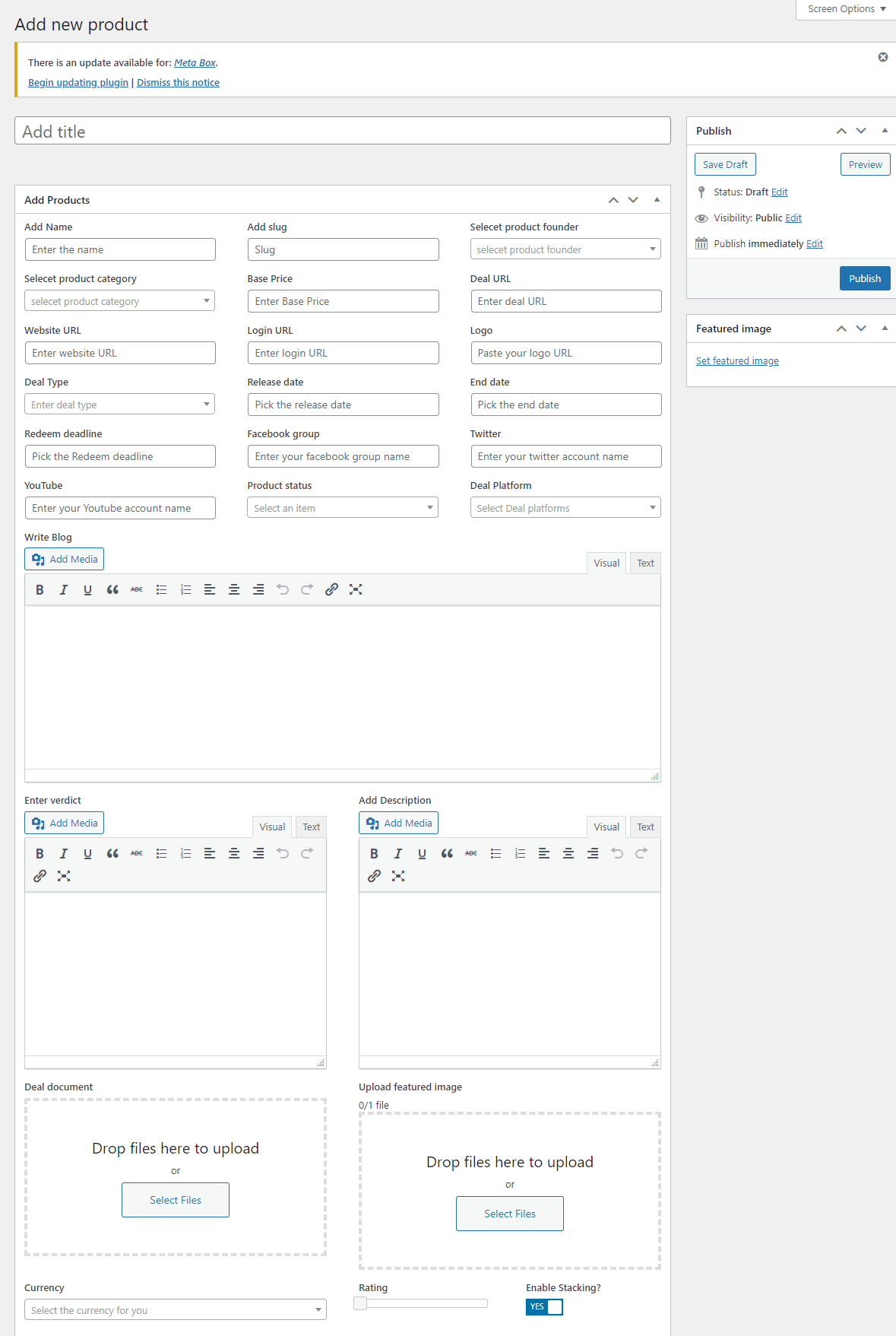
* Installation procedure is very easy
* XAMPP is available for Windows and Linus
* It has several modules
* Offers both a full and a standard version [16]

# 

# 6.6 Snapshots

## 6.6.1 Product Add

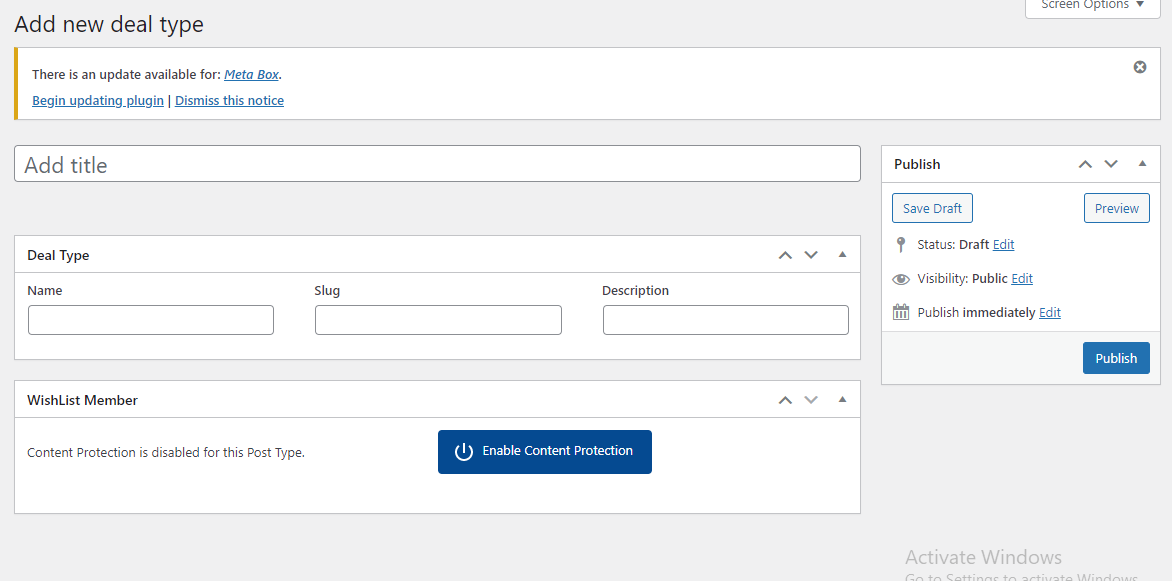
Admin can easily add product to our system by providing their name, product\_founder\_id, product\_category\_id, name, slug and their membership type. After successful form submission, the admin will check the product information.



**Figure 6.1. Admin Product Add.**

## 6.6.2 Add new Deal Type

Admin can easily create new deal type and also edit deal type on our system.

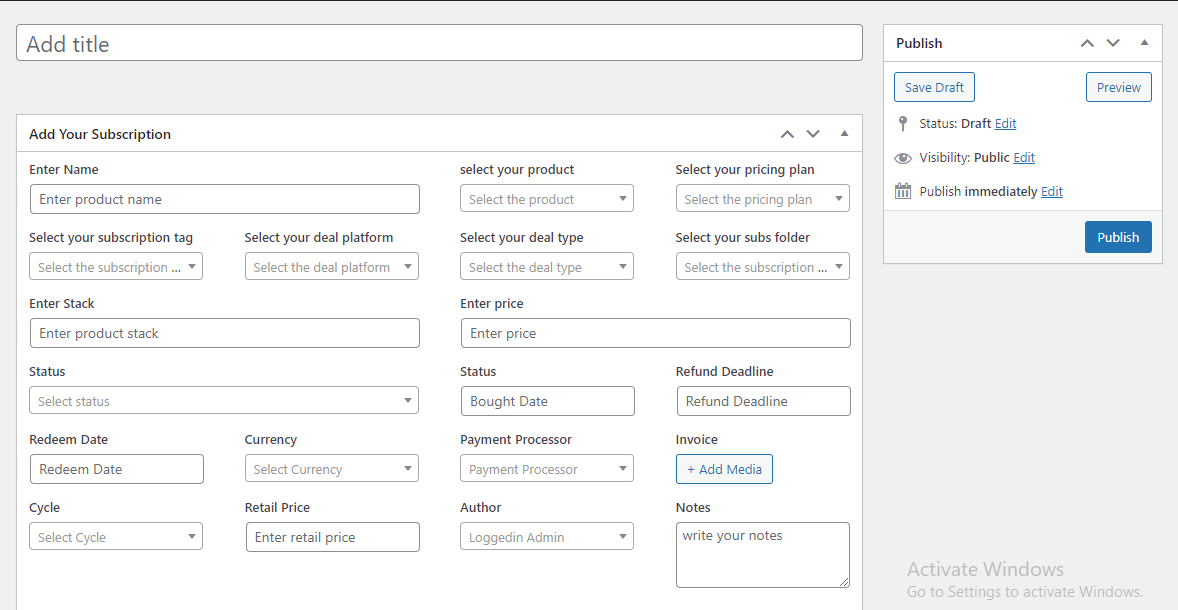


**Figure 6.2. Add new deal type.**

## 

## 6.6.3 Subscription Add

Admin can easily add subscription into our system. All other users are able to see all the subscription. Users are also able to buy or delete their subscription.

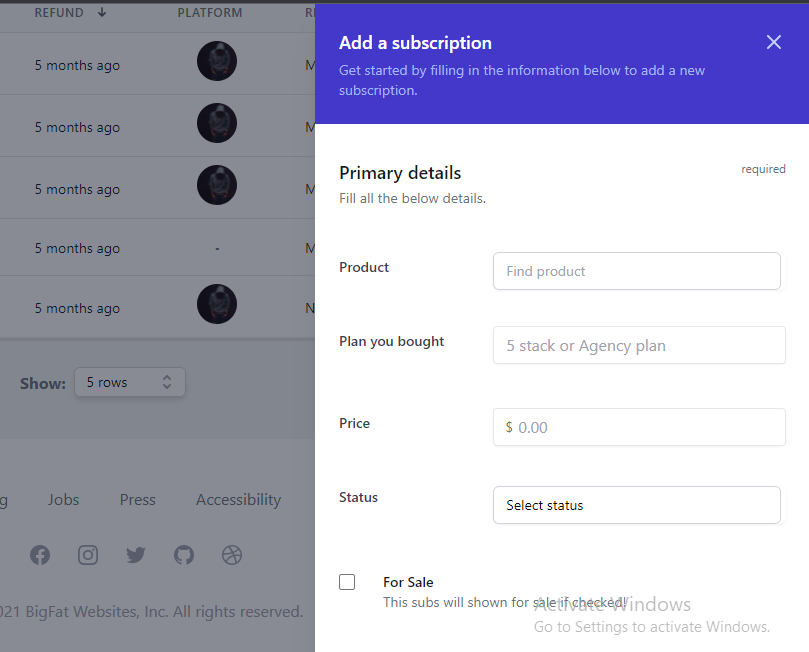


**Figure 6.3. Subscription Add.**

## 

## 6.6.4 Add Subscription by User

User can easily add subscription. Users are able to receive notifications.

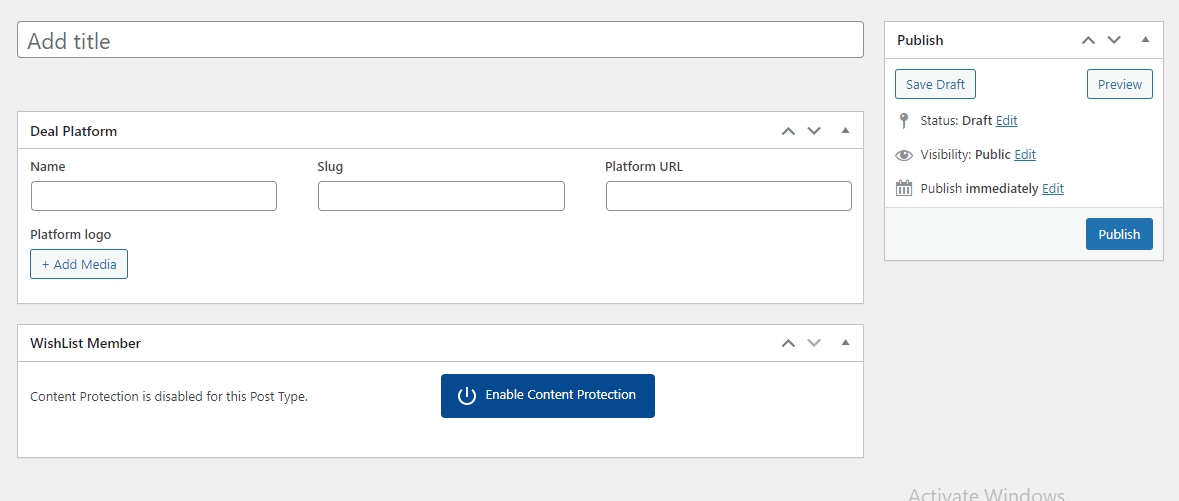


**Figure 6.4. Subscription Add By User.**

## 

## 6.6.5 Deal Platform

Admin can easily create deal platform and also edit deal platform.

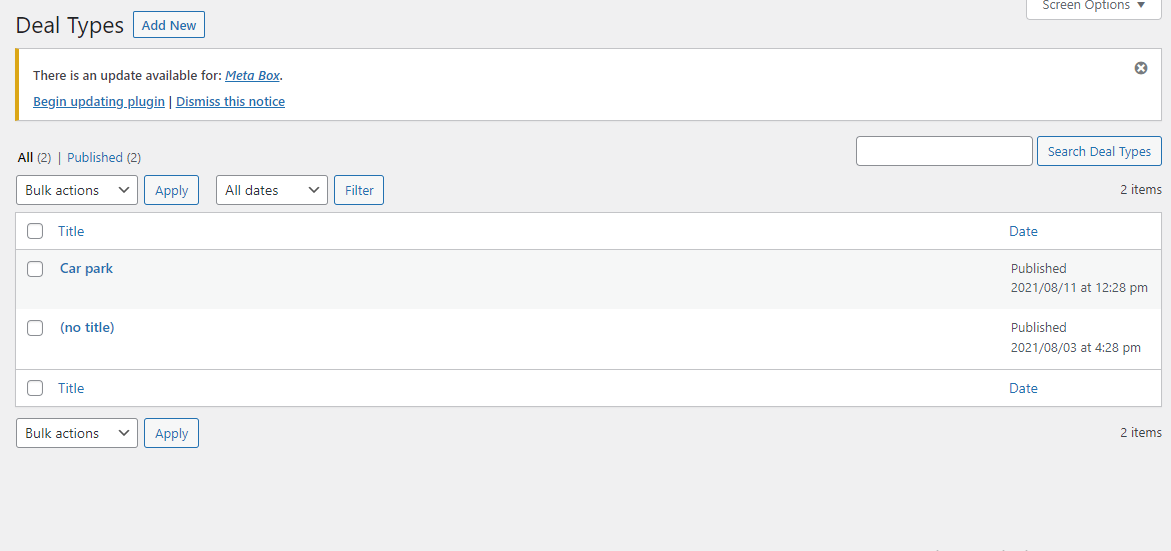


**Figure 6.5. Deal Platform.**

## 

## 6.6.6 Deal Type

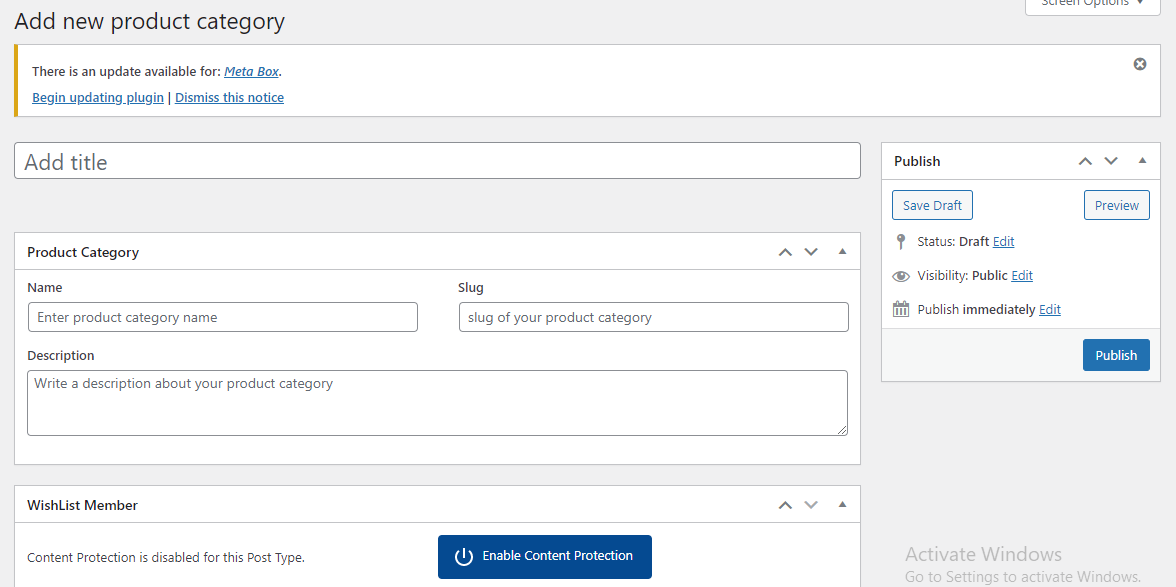
Admin can add deal type which subscription user want to buy.



**Figure 6.6. Deal Type.**

**6.6.7 Product Category**

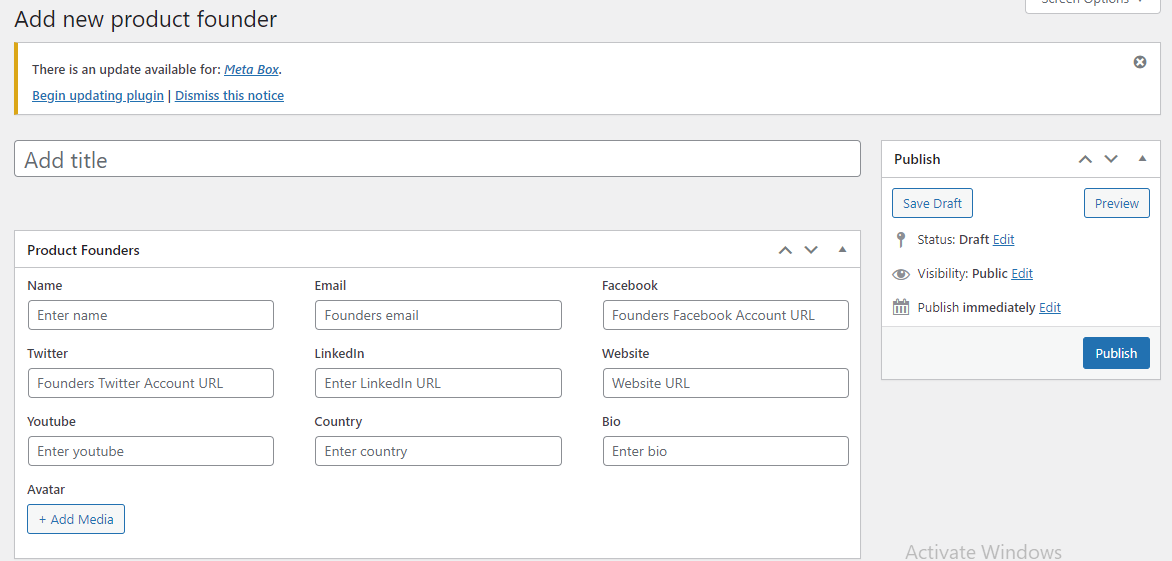
Admin can easily add category to our system by providing their name, slug and description. After successful form submission, the admin will check the product category.



**Figure 6.7. Product Category.**

**6.6.8 Add New Product Founder**

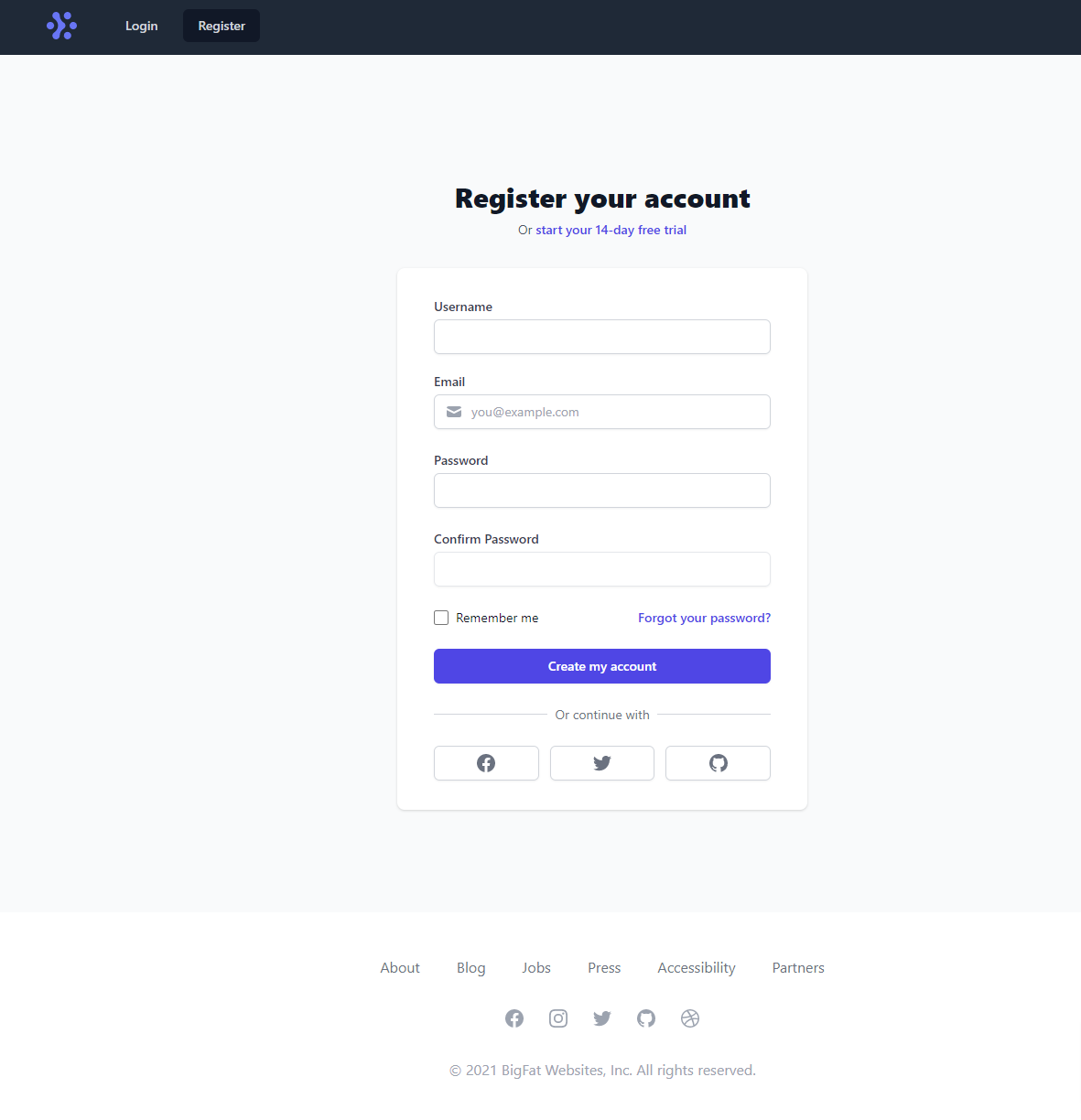
Admin can easily add new product founder to our system by providing their name, email, Facebook, Twitter, country and Avatar. After successful form submission, the admin will check the product founder.



**Figure 6.8. Product Founder.**

**6.6.9 User Register**

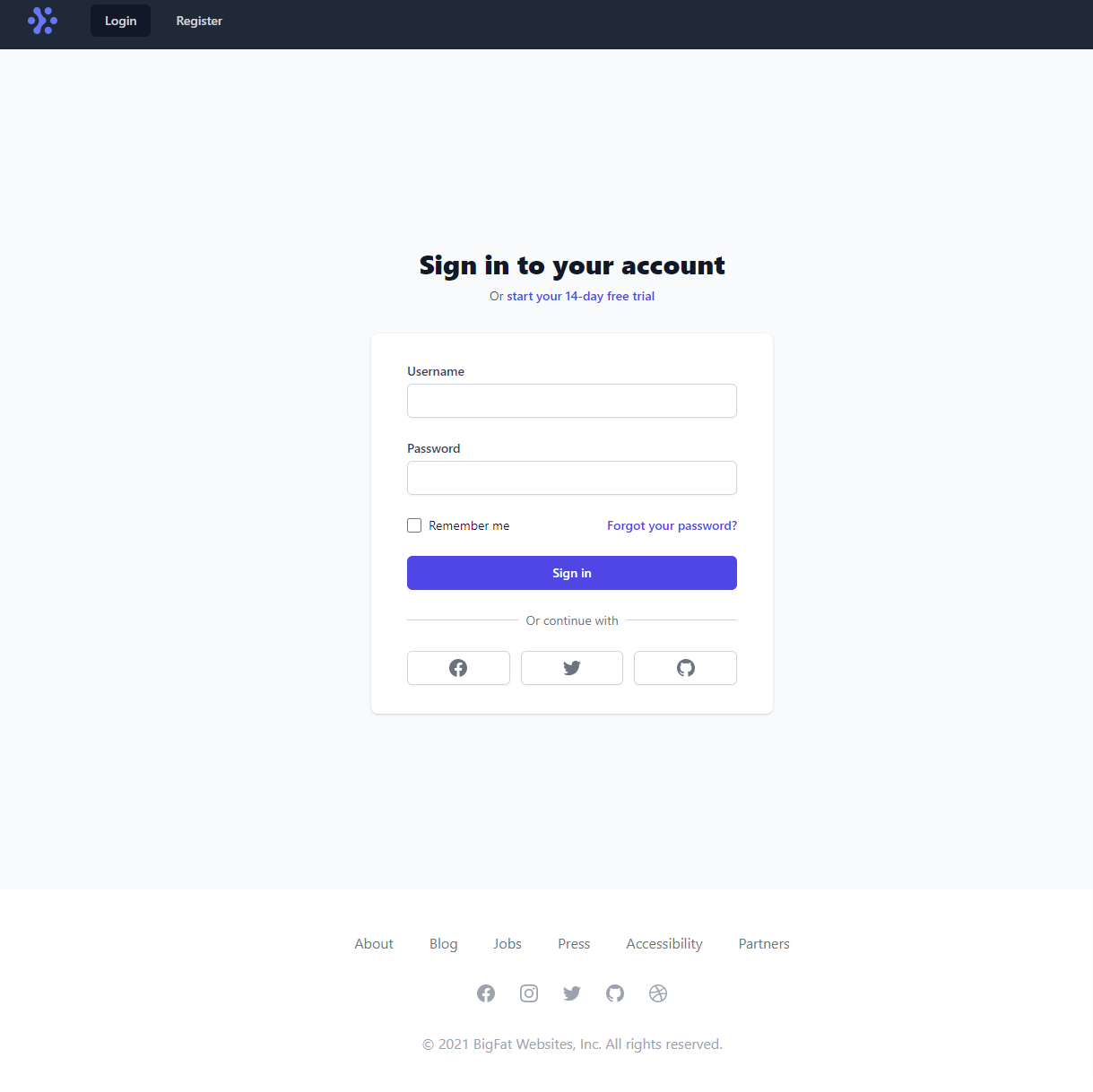
User can register on our system, user can easily signup.



**Figure 6.9. User register.**

**6.6.10 User Login**

User can login on our system, user can easily sign in.

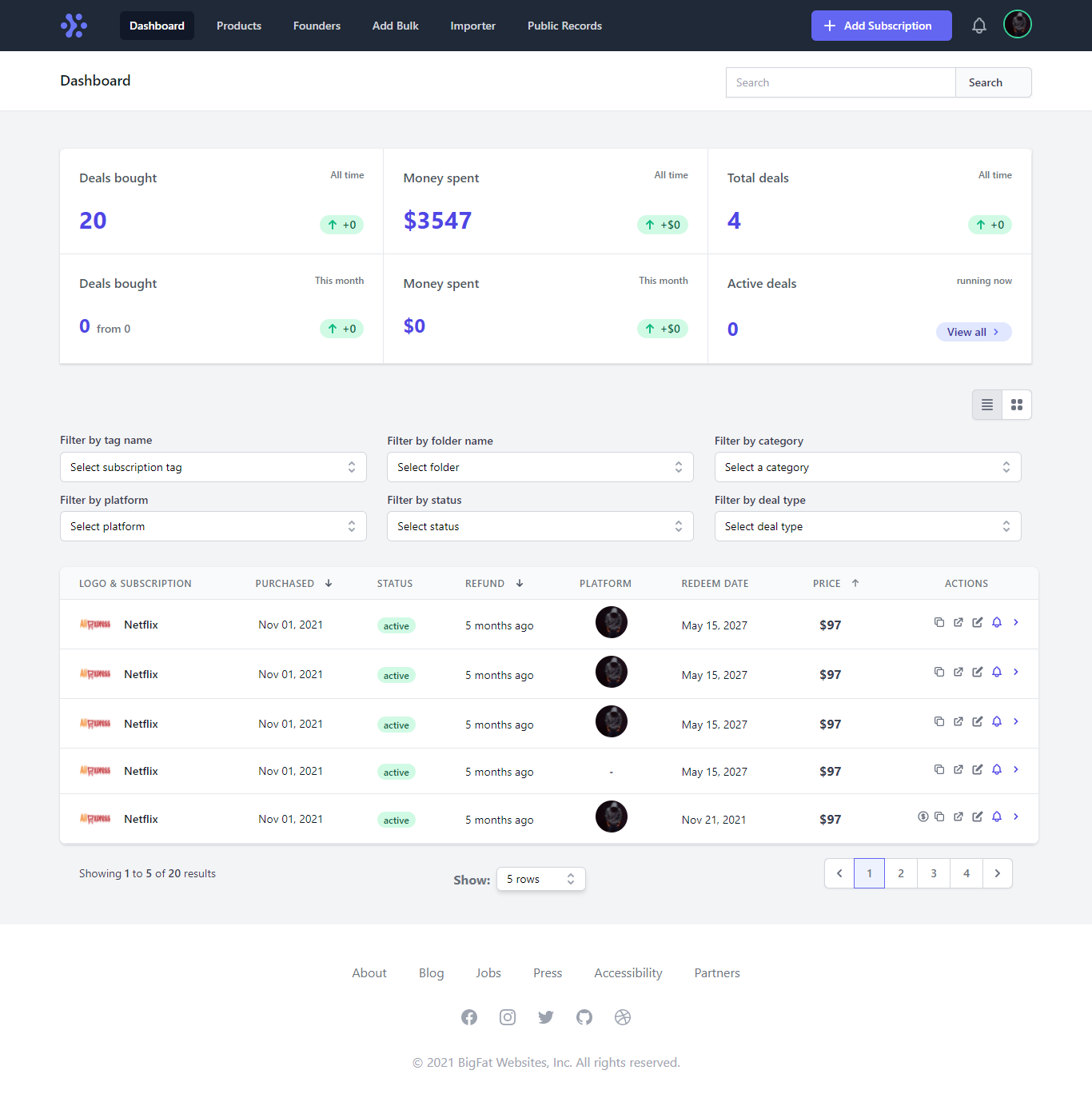


**Figure 6.10. User login.**

**6.6.11 User Dashboard**

In user dashboard, user can easily see –

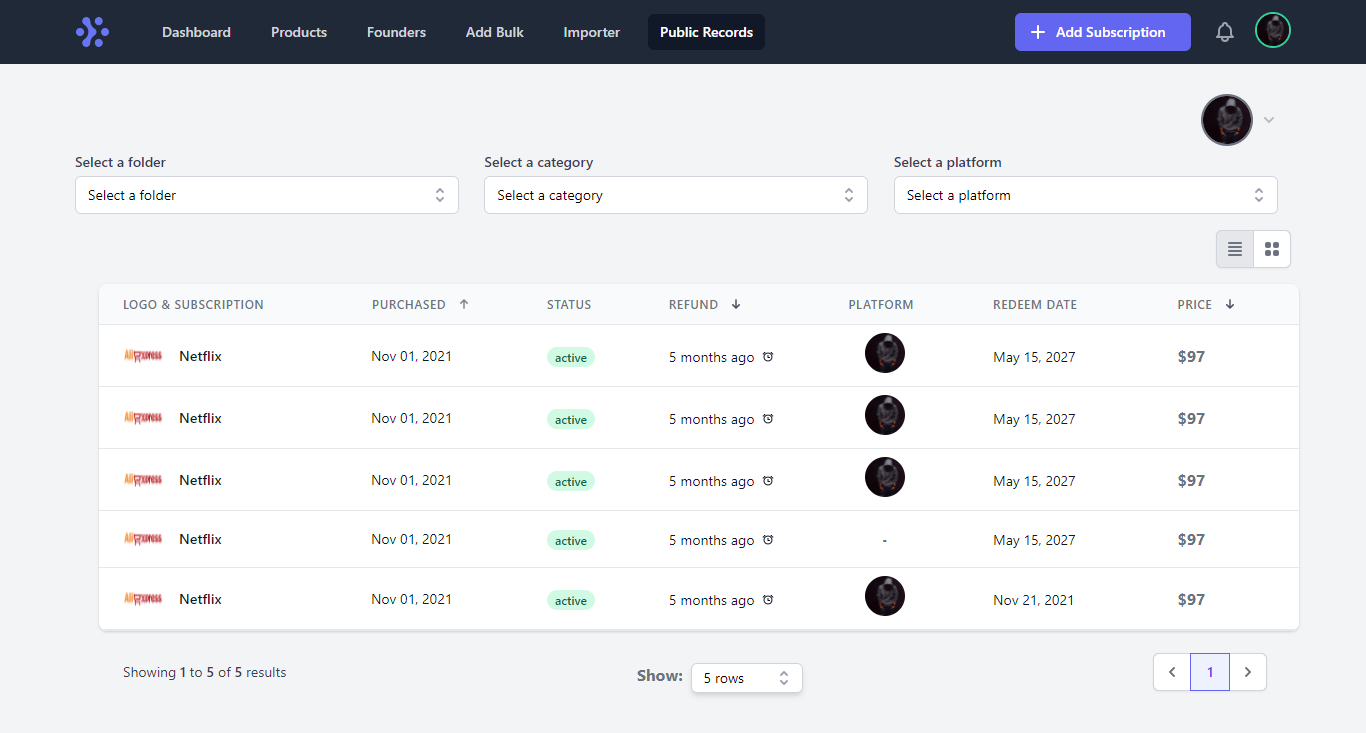
1. Total number of users in our system.
2. Total number of subscriptions.
3. Total number of subscription types.
4. Total number of requests to update membership.
5. Total number of subscriptions sold.
6. Total number of active deals.
7. Total numbers of deals bought.
8. Total amount of money spent.
9. Total numbers of active deals.



**Figure 6.11. User dashboard.**

**6.6.12 All Subscription in User Dashboard**

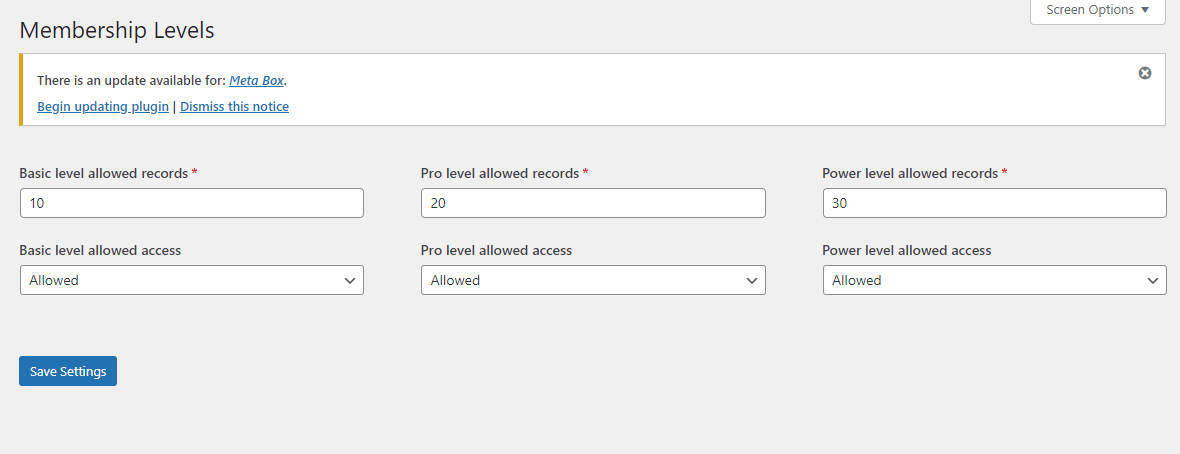
By clicking into the All-Users tab, User will see a list of subscription with their information. User can filter as well by clicking into the top 3 tab to see folder, category and platform for see specific subscription.



**Figure 6.12. All subscriptions.**

**6.6.13 Membership Level**

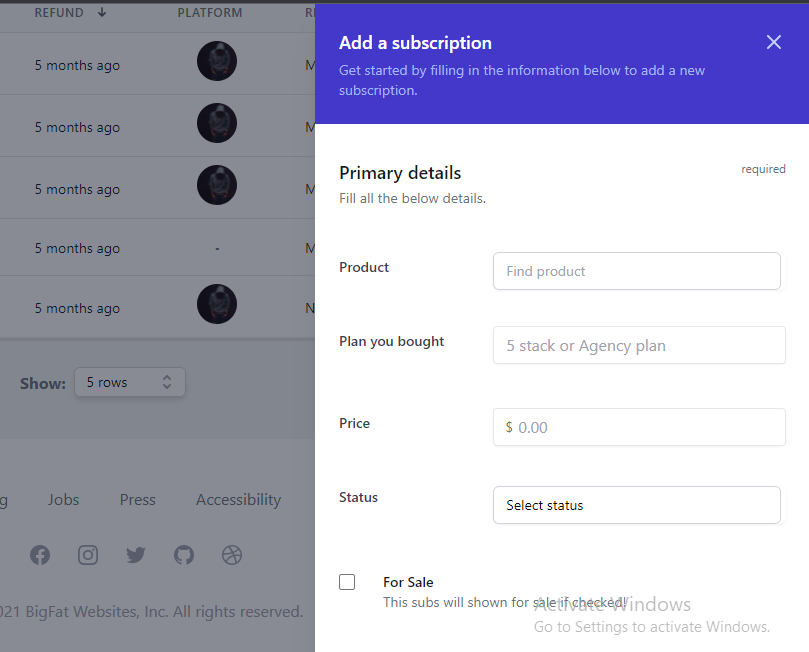
User can get a membership plan on our system.



**Figure 6.13. Membership Level.**

**6.6.14 User Subscription Add**

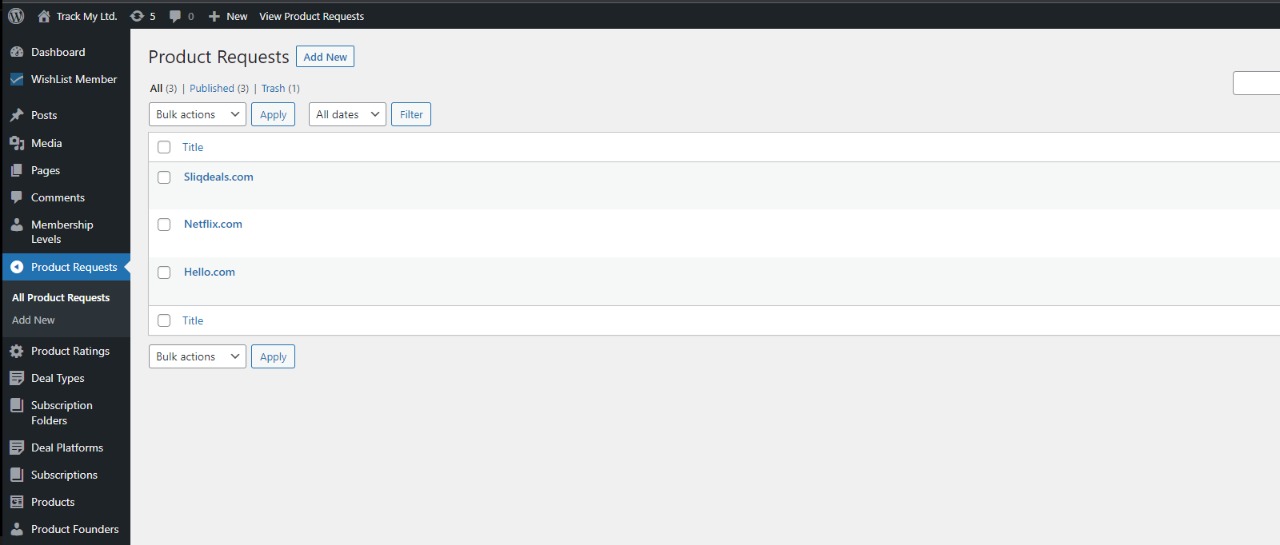
User can easily add subscription on our system.



**Figure 6.14. User Subscription Add.**

**6.6.15 User Product Request**

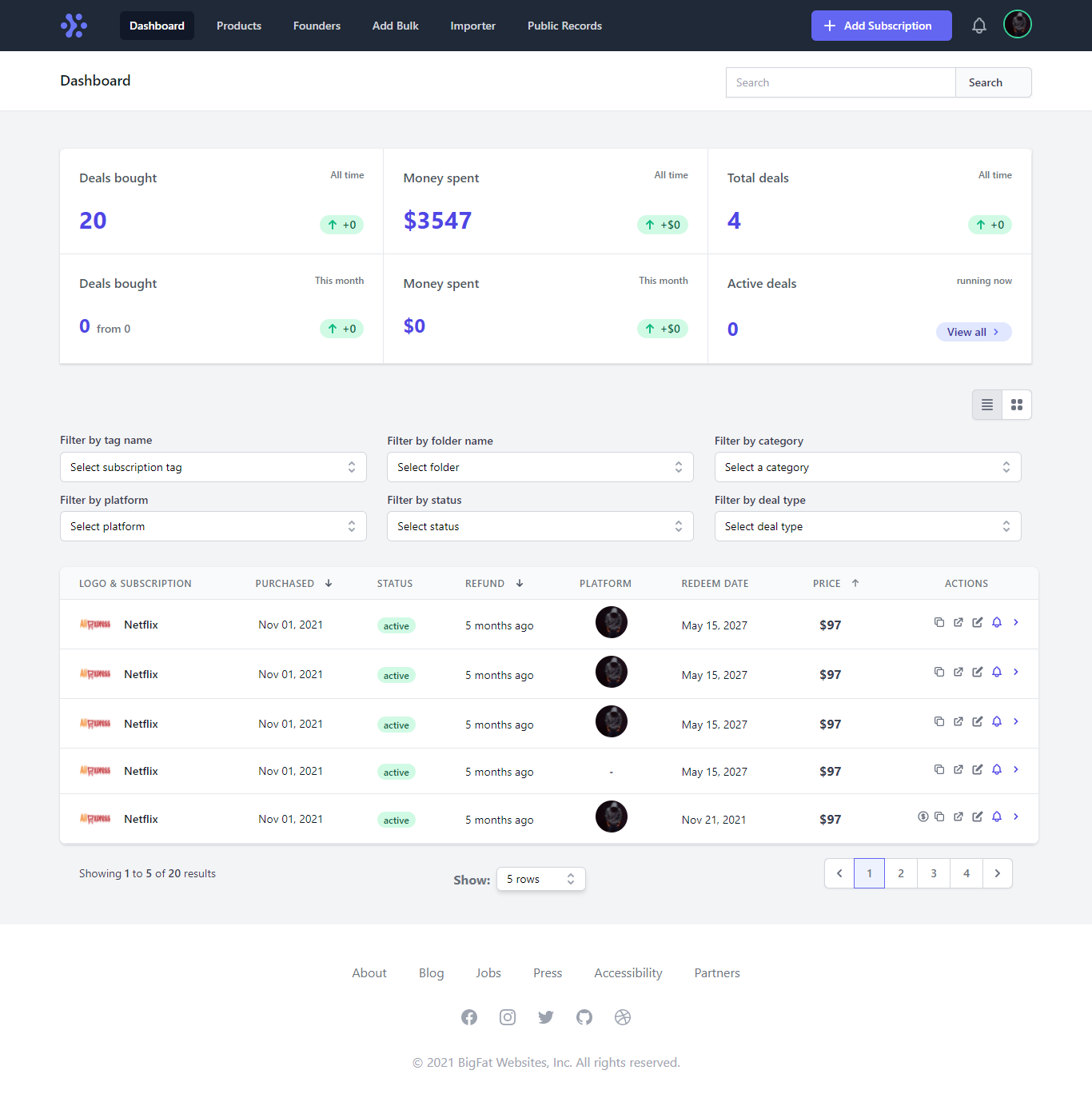
User can request product on our system.



**Figure 6.15. Product Request.**

**6.6.16 User Dashboard after Purchase**

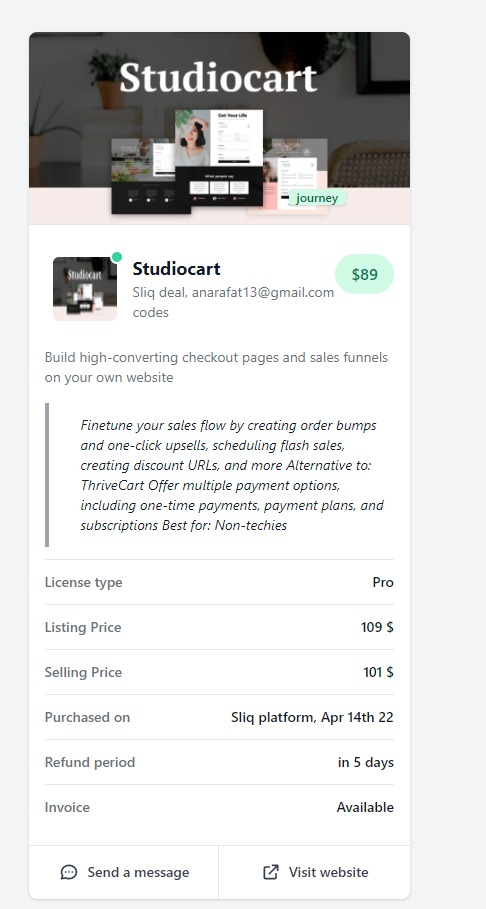
User can see updated dashboard on this system when they purchase a subscription.



**Figure 6.16. Dashboard after purchase.**

**6.6.17 Purchase Subscription Information**

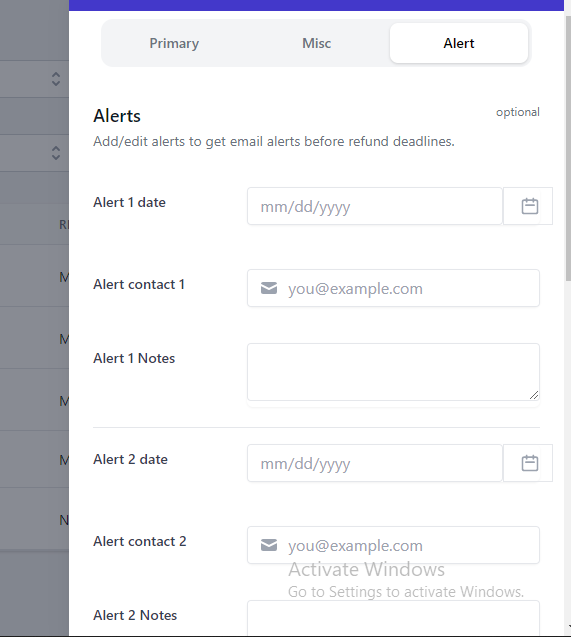
User can see purchase subscription information on this system.



**Figure 6.17. Purchase Subscription Information.**

**6.6.18 Subscription Alert**

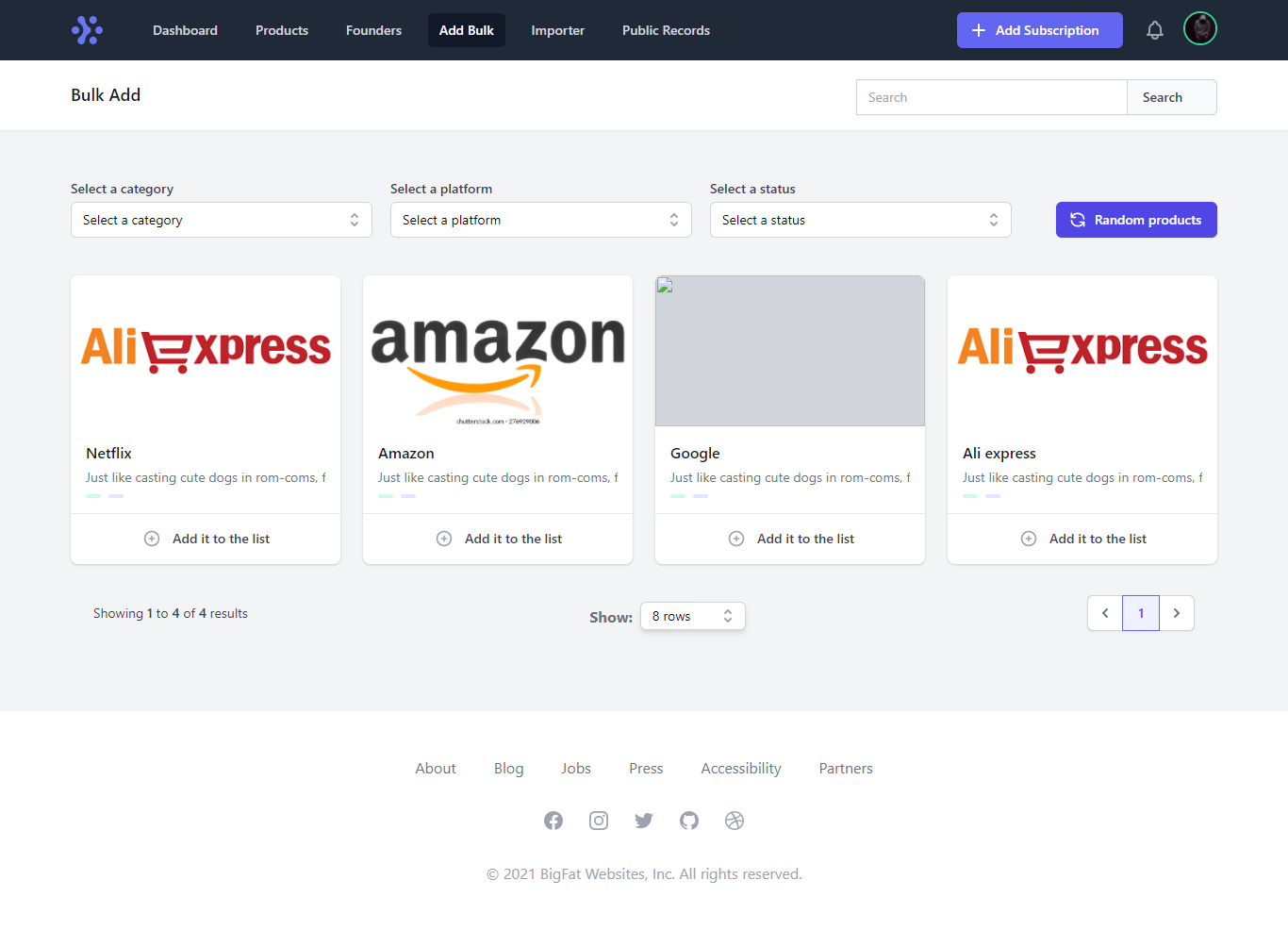
User will receive notification alert for subscription end period date on this system.



**Figure 6.18. Subscription alert.**

**6.6.19 Subscription Bulk Add**

User can add bulk subscription on this system.



**Figure 6.19. Subscription Bulk Add.**

**6.7 Summary**

The implementation of this project is discussed in this chapter. What technology did we utilize to create this website. We also go over each feature with screenshots.

# Introduction

**Chapter 7 DISCUSSION AND CONCLUSION**

The final goal of my project is to develop a website for general people so that with the system anyone can be vaccinated. The system was successfully completed. After completing this project, i want state that i did my best to design this system in the most appropriate, useful, and user-friendly manner possible. I achieve the system's expected outcomes. The system is ready to use from anywhere, on any PC, and it can be accessed via the internet from any computer.

# Discussion

This project is broken down into various sections. Introduction, literature study, and methodology, software requirements specification, normalization, and data dictionary, software design, and implementation are the items on the list.

I covered the project's goals, an overview of the current system, and a proposed system in the introduction.

I examined the software development life cycle and which software model is utilized in the system in the literature review and methodology.

I discussed the software development life cycle and which software model is utilized in the system in the literature review and methodology.

I learned about the requirements of a system that are needed by the end-user by looking into software requirements specifications.

I normalized several tables in the normalization and data dictionary section to handle data redundancy by reducing multiple value, partial dependency, and transitive dependency.

I've focused on the ER diagram, activity diagram, and user case diagram in software design.

I described the implementation tools that we used during the construction of our system in implementation.

# Limitation of the system

Currently in my project admin maintains this system. There is no panel for users. User can’t take appointment for their vaccination from their end. User can’t freely communicate with the owner of the vaccination center. The project is only web based.

# Future work

1. I will implement appointment system so that any people can take an appointment for vaccination.
2. I will develop a chat system so that people can communicate easily for their vaccination.
3. I will add a mobile app for my project.

# Summary

In this chapter I have discussed the limitations of my project and the future plan of my project.

# References

[1] <https://www.truebill.com/feature/manage-subscriptions> [Accessed on: 31 March, 2022].

[2] [https://money.usnews.com/money/personal-finance/saving-and-budgeting/articles/track-and- manage-subscriptions-with-these-apps](https://money.usnews.com/money/personal-finance/saving-and-budgeting/articles/track-and-%20%20%20%20%20manage-subscriptions-with-these-apps) [Accessed on: 31 March, 2022].

[3] [https://appsumo.com/deals/subscription-tracker-and-management-notion/](https://appsumo.com/products/subscription-tracker-and-management-notion/) [Accessed on: 31 March, 2022].

[4] <https://www.guru99.com/what-is-php-first-php-program.html> [Accessed on: 1 April, 2022].

[5] <https://www.techtarget.com/searchcontentmanagement/definition/WordPress> [Accessed on: 1 April, 2022].

[6] [https://community.codenewbie.org/lanrewaju/100daysofcode-chapter-1-html-and-css-basics-3 709#:~:text=What%20is%20HTML%3F,for%20web%20 pages%20and%20applications](https://community.codenewbie.org/lanrewaju/100daysofcode-chapter-1-html-and-css-basics-3709#%3A~%3Atext%3DWhat%20is%20HTML%3F%2Cfor%20web%20pages%20and%20applications). [Accessed on: 1 April, 2022].

[7] <https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS> [Accessed on: 1 April, 2022].

[8] <https://en.wikipedia.org/wiki/JavaScript> [Accessed on: 1 April, 2022].

[9] <https://v2.vuejs.org/v2/guide/?redirect=true> [Accessed on: 1 April, 2022].

[10] [https://www.siteground.com/tutorials/php-mysql/mysql/#:~:text=MySQL%20is%20a%20fre ely%20available,ease%20and%20flexibility%20of%20use](https://www.siteground.com/tutorials/php-mysql/mysql/#%3A~%3Atext%3DMySQL%20is%20a%20freely%20available%2Cease%20and%20flexibility%20of%20use) [Accessed on: 1 April, 2022].

[11] [https://www.geeksforgeeks.org/introduction-to-tailwind- css/#:~:text=Tailwind%20CSS%20is%20basically%20a,have%20to%20fight%20to%20override](https://www.geeksforgeeks.org/introduction-to-tailwind-%20%20%20css/#:~:text=Tailwind%20CSS%20is%20basically%20a,have%20to%20fight%20to%20override) [Accessed on: 1 April, 2022].

[12] <https://www.toppr.com/ask/question/web-server/> [Accessed on: 1 April, 2022].

[13] <https://www.wpbeginner.com/glossary/apache> [Accessed on: 1 April, 2022].

[14] <https://www.webopedia.com/definitions/browser/> [Accessed on: 1 April, 2022].

[15] <https://code.visualstudio.com/docs> [Accessed on: 1 April, 2022].

[16] <https://en.wikipedia.org/wiki/XAMPP> [Accessed on: 1 April, 2022].