TRADE TUNNEL

PROJECT REPORT

CSE 686 – INTERNET PROGRAMMING

PROF. EDMUND YU

Documentation provides detailed information on system architecture, system components, features and database system

TABLE OF CONTENTS

Table of Contents

TRADE TUNNEL

Problem Statement

To develop a Trading web application which allows the users to seamlessly buy and/or sell their items using this platform.

INTRODUCTION

It is a common practice to give away or scrap the items which you don't need anymore. How about selling them instead? This web application gives a real - time opportunity to meet potential buyers of such products. It's an attempt to bridge the gap between the buyers and sellers who would want to sell and/or buy items online.

MOTIVATION

It is not uncommon to notice students posting advertisements on social media pages of products they wish to sell or give away. Sellers often face problems meeting the potential and genuine buyers of the products. This Web Application is a software solution to address this issue as it serves as a mediator between the sellers and the buyers.

Original Idea hovers around to create a platform for SU students to trade in the products they wish to. However, we decided to go beyond SU boundaries and allow anyone to use the service who wishes to trade.

TEAM MEMBERS

1. Himanshu Chhabra (Team Lead) - 777993851
2. Chetali Mahore - 750500177
3. Krupa Mavani - 351960088
4. Aarsh Patil - 329989968
5. Sowmya Padmanabhi - 518645655
6. Anagha Fatale - 438039600
7. Vinu Kundnani - 470316255
8. Sonali Ratnam - 600148631

1. Features

This section focuses on several features that that have been developed.

1.1 REGISTER / LOGIN FEATURE

- The web application supports sign in feature
- The user can sign in using the registered credentials as shown in the figure 1.1.1
- The web application allows the user to get itself registered to the website as shown in figure 1.1.2

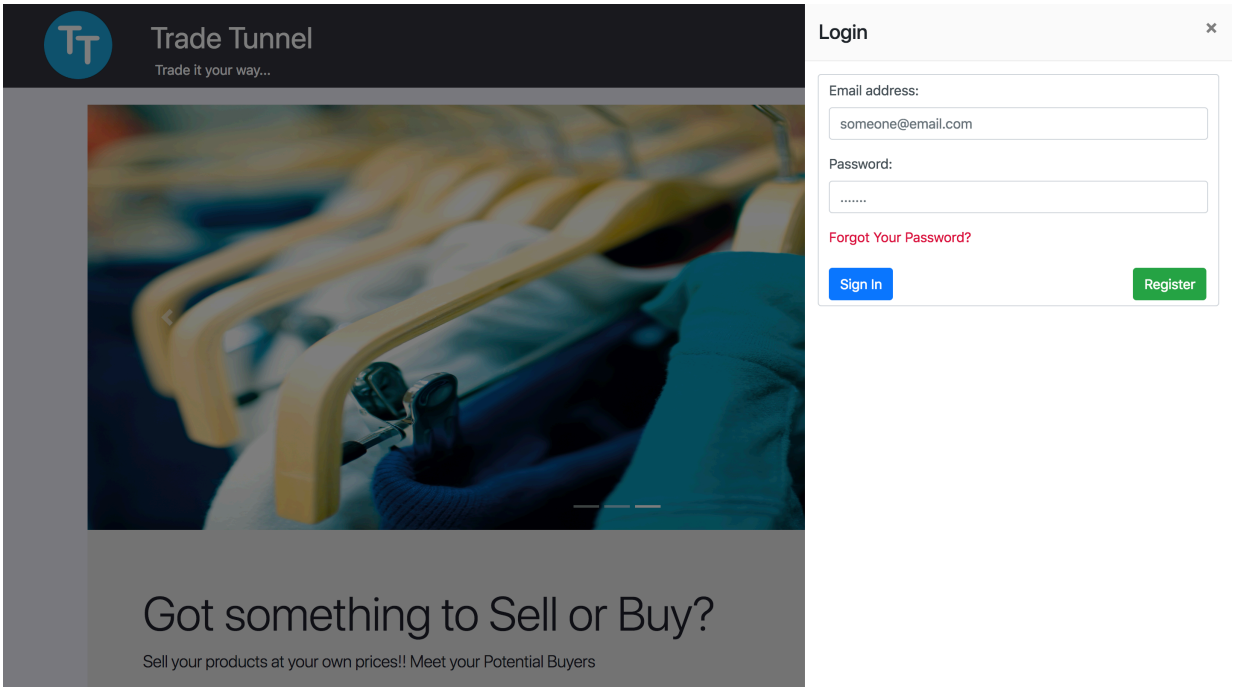


Figure 1.1.1

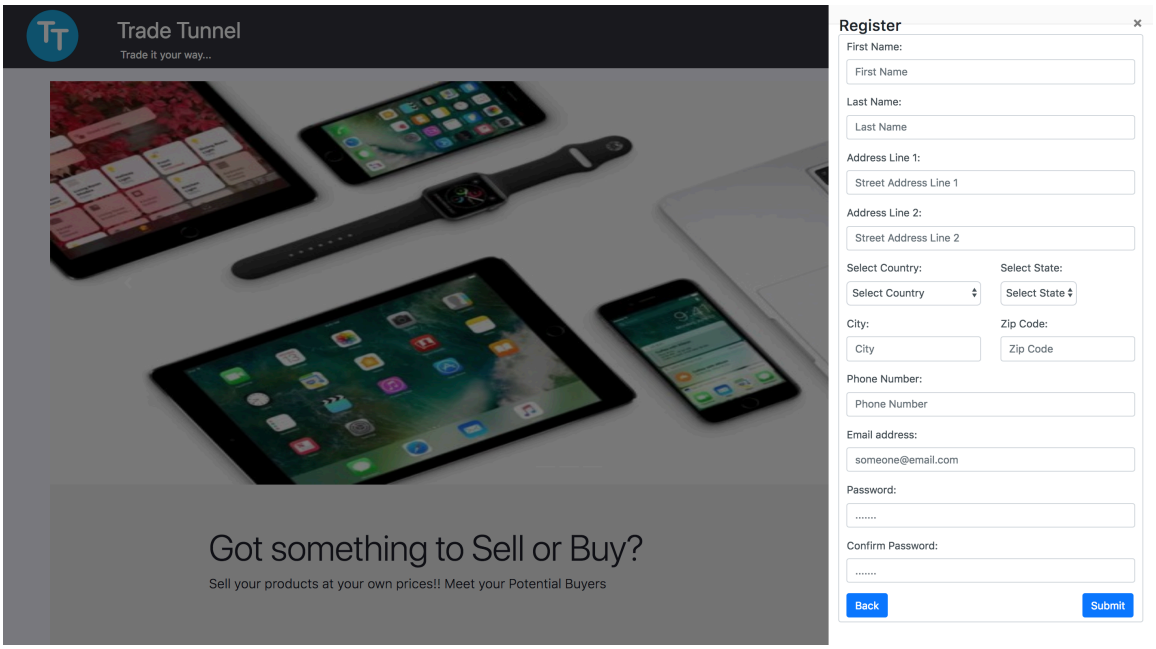


Figure 1.1.2

1.2 ADVANCE CUSTOM SEARCH FEATURE

- Custom Search assists the user to search for the products on the website using keywords as shown in the figure 1.2.1
- This feature enhances the user experience by allowing the users to search for products based on popular keywords e.g.: laptop, Dell, iPhone, shoes, red dress, table etc.
- The system uses spring query language support to return the search results in a very efficient manner.
- The user can browse through the searched products and click on any product of its choice to view product details
- As seen from the figure 1.2.1 below, the Search results returns all the products that has a matching substring either with the product name or product description.

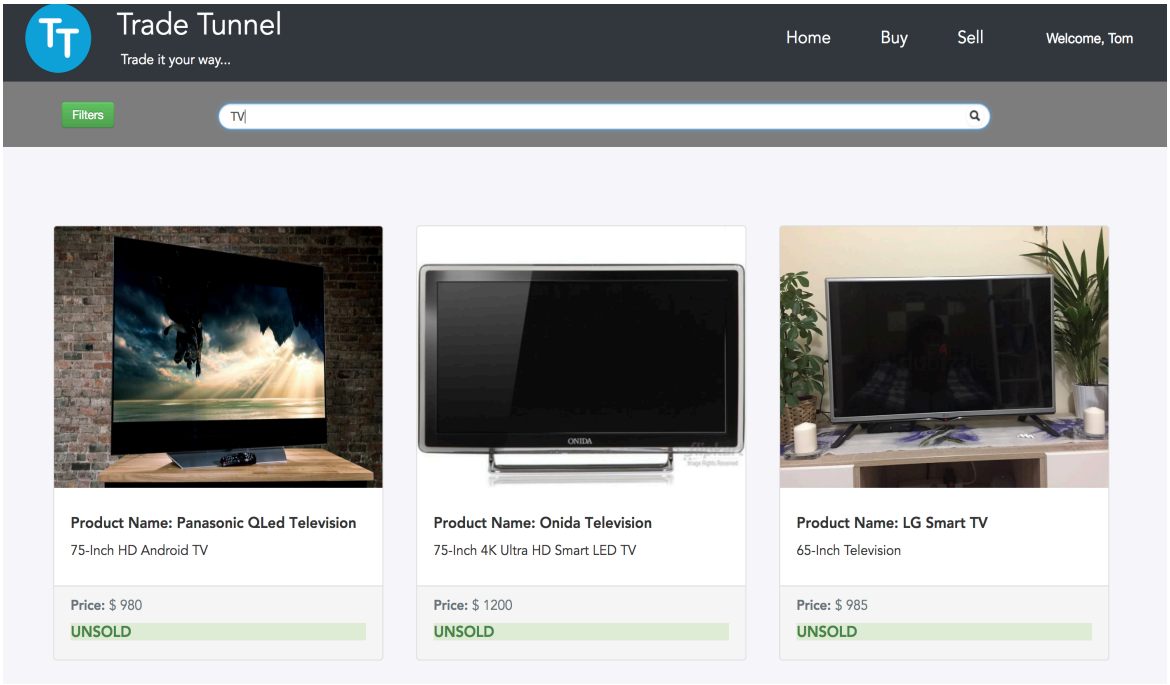


Figure 1.2.1

1.3 PAGINATION FEATURE

- Application provides an elegant view to browse across several products on the system as shown in figure 1.3.1
- On successful login into the application, the user can choose to visit the Buyer's page.
- The application provides the user with an exhaustive list of products with 9 products listed on each page.
- The user can navigate across the pages using the next and previous buttons on the bottom of the page as shown in figure 1.3.2
- The buyers view provides an ease access to filter products feature and search products feature all on the single page as shown in figure 1.3.1

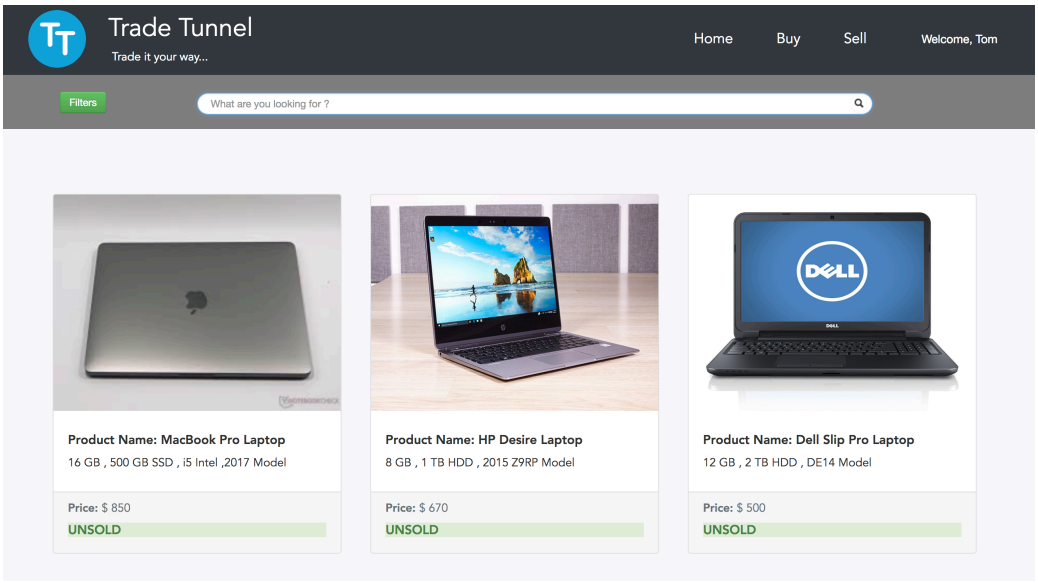


Figure 1.3.1

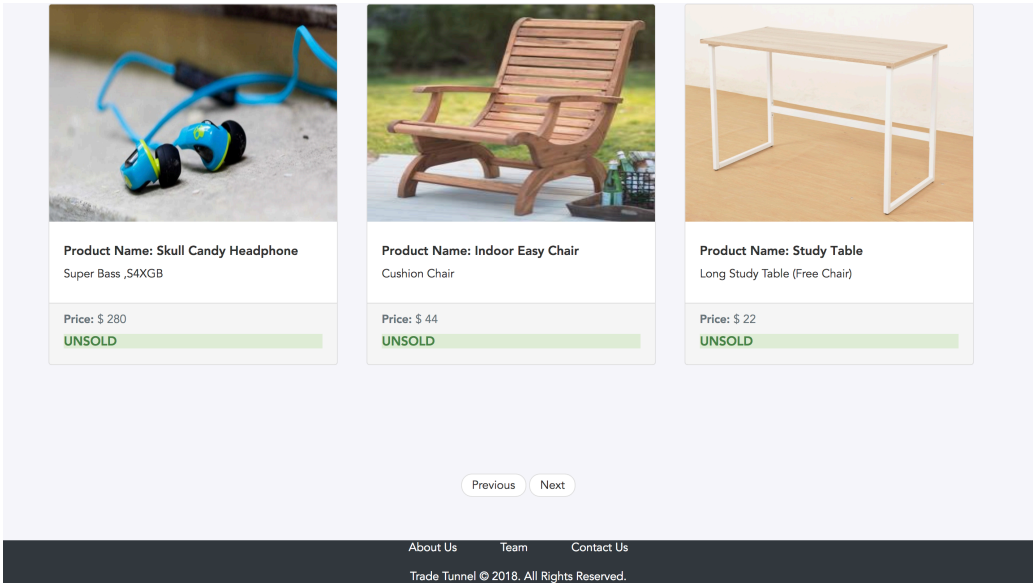


Figure 1.3.2

1.4 PRODUCT DETAILS FEATURE

- On clicking on a product, the user is redirected to the product details page as shown on in figure 1.4.1

- User can browse through images of the product which are uploaded by the seller
- The application provides product details which includes product name, description and prospective price set by the seller
- Application provides sellers information as shown in figure 4.1, assisting the buyer to proceed with the purchase by contacting the seller.

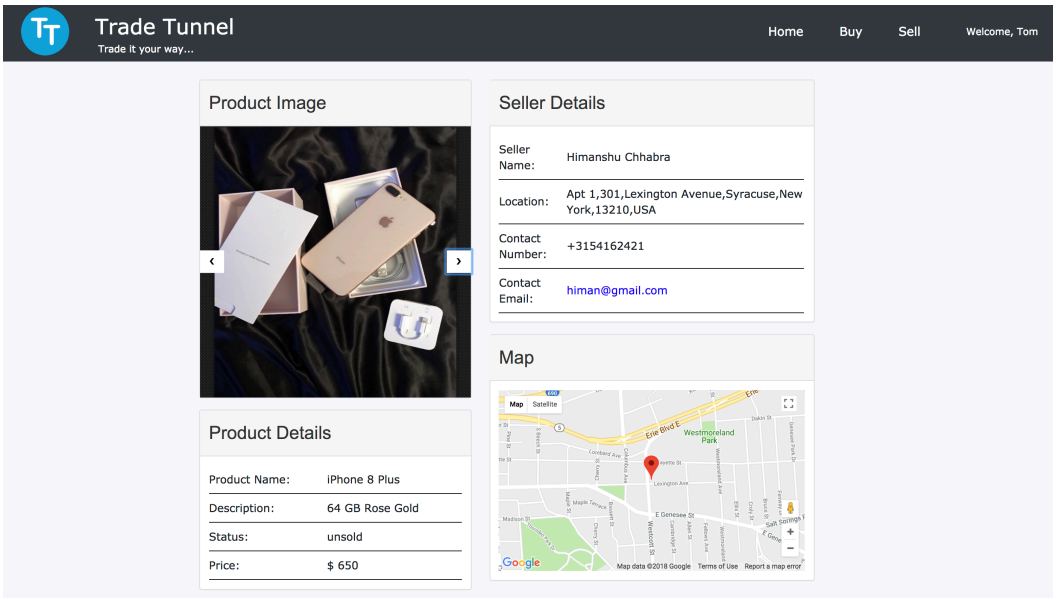


Figure 1.4.1

1.5 I'M INTERESTED (NOTIFICATION / QUICK EMAIL FEATURE)

- The Application provides as easy notification feature
- The buyer can simply click on the email id of the seller as on the product details page shown in figure 1.4.1.
- This action launches the default configured email application with the to: field auto populated with the seller's email id.

1.6 LOCATE ME THE SELLER FEATURE

- The Application provides a map on the products details page as shown in the figure 1.4.1
- The map marks the seller on the google maps using sellers address, the buyer can click on the maps.

- This action will redirect the user to the google maps thereby assisting the user to select the best route to the seller of the product.

1.7 FILTER PRODCTS FEATURE

- The Application allows the user to filter the products using a variety of filter options which work in conjunction with each other as shown in figure 1.7.1
- User can filter the products using several options as listed below
  - Categories - allows the user to filter products based on categories
  - Sub-categories - allows the user to filter products based on sub-categories, sub categories are populated specific to the categories selected.
  - Price range - allows the user to filter products within the price range.
  - Location based service - allows the users to view products at a particular location.
  - These filters can be used in conjunction providing better user experience.
  -

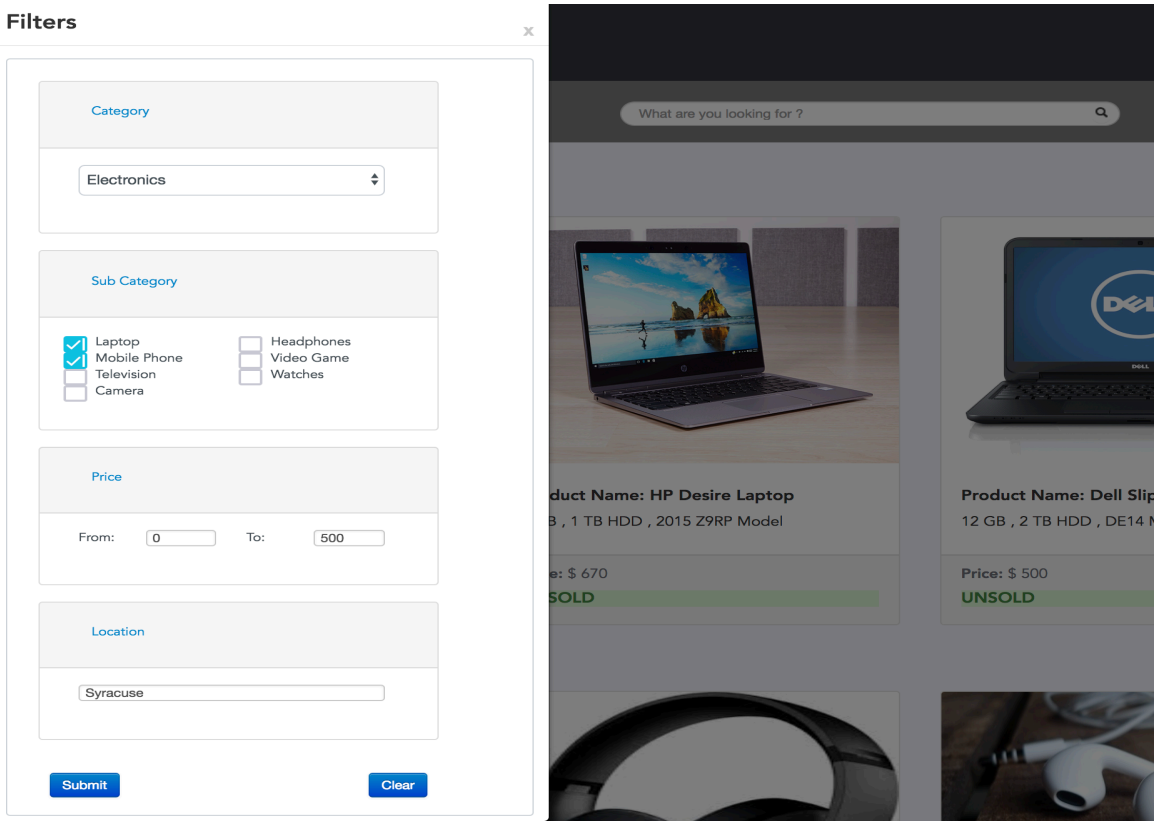


Figure 1.7.1

1.8 PRODUCTS ADVERTISEMENT FEATURE

- User can navigate to the seller's page on successful login either through the home page or using the navigation bar.
- Application allows the user to sell any product online, user has the freedom to advertise the product as the user wishes to.
- Application provides an opportunity to the user to provide a relevant product name along with a brief description of the product for sale.
- User can categorize and sub-categorize their products, this is used to display the seller's products during search or filter operations performed by the buyer on buyer's page.
- Application allows the user to upload several product images as shown in the figure 1.8, the user Tom has uploaded 2 images for the product MacBook Pro.
- Reset button will clear all the fields of the form assisting the user to redo the complete form if required.

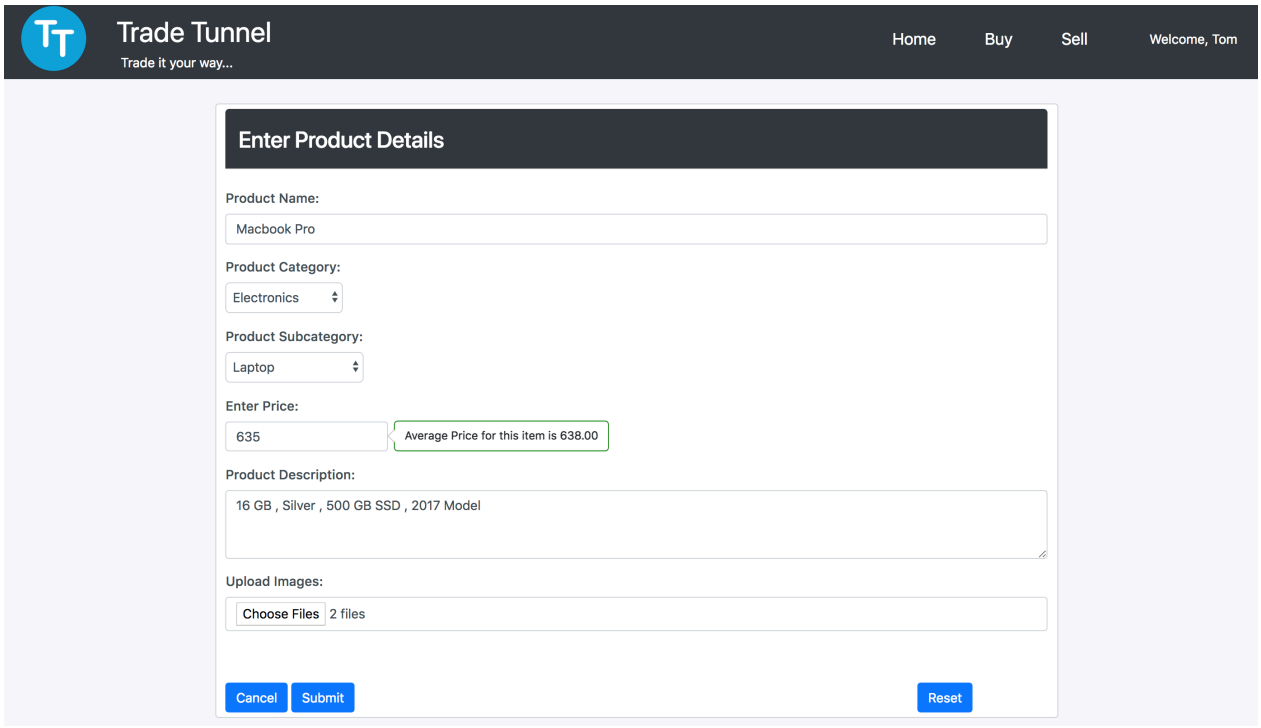


Figure 1.8.1

1.9 PRODUCT PRICE ASSISTANCE FEATURE

- As the seller's page allows the user to advertise their products, the price assistance feature helps in setting the right price for the product.
- The Application assists the user to set a competitive price for the product by providing with an average price of all products which fall under the selected subcategory as shown in figure 1.8.1

1.10 USER PROFILE AND PRODUCT HISTORY

- Application maintains the user's profile information as well as products history
- User can navigate to the Account's page and view its personal information as well as browse through its product history as shown in figure 1.10.1, on successful login attempt.

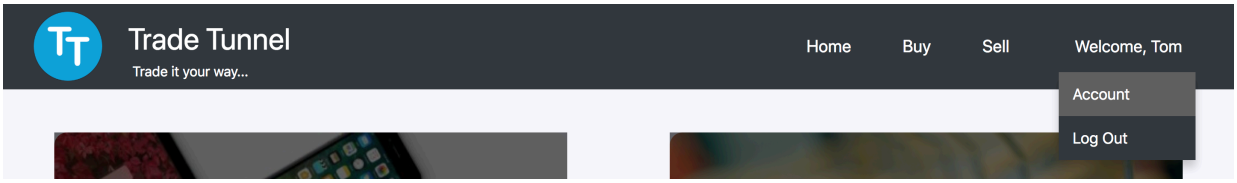


Figure 1.10.1

- On selecting my profile option as shown in figure 1.10.2, the user can view and/or update its personal information by clicking on the edit button.
- On selecting my products option as shown in figure 1.10.3, the user can view all its products uploaded for sale on the left panel.
- User can click on any of its products and view/update the product information.

1.11 PRODUCT STATUS TOGGLE FEATURE

- Application provides with a simple toggle button on my products section as shown in figure 1.10.3, to toggle the product status between sold and unsold product status.
- Simple toggle enhances the user experience as the user is not required to perform additional steps just to change the product status.

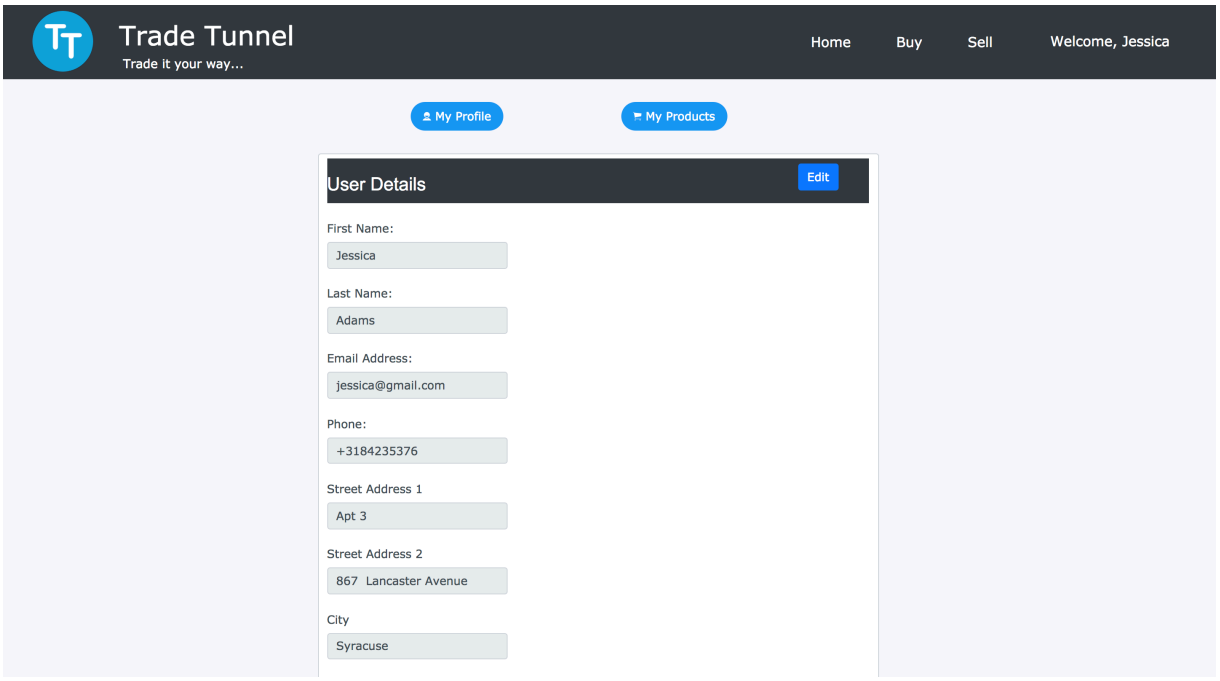
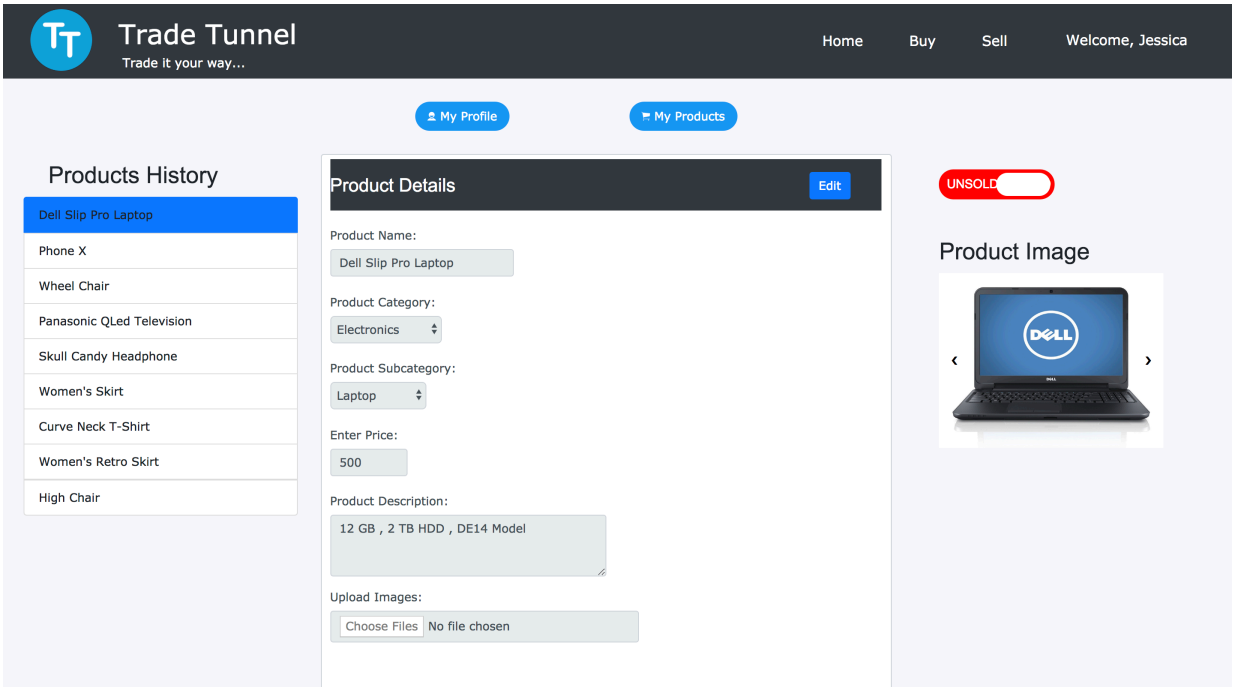


Figure 1.10.2



1.10.3

2. System Design and Architecture

This section provides detailed information on system design, system architecture and technologies used to develop the web application

2.1. SYSTEM DESIGN OVERVIEW

- The Software system follows the client server model for communication.
- Software system is mainly divided into three components - the client side application, the server side application and the core database.
- The server side is implemented using Spring Data REST and Spring Boot JAVA frameworks.
- The client side is implemented using JavaScript, JQuery, AJAX, HTML5, CSS3 and Bootstrap 4.
- System follows Micro-services architecture paradigm which is a modern web architecture trend.
- Client – Server communication is fulfilled using the HTTP protocol.
- JSON is used as the data exchange standard between the client application and the server.

2.2. SYSTEM ARCHITECTURE

- The system architecture is as shown in the figure 2.1.1
- The software is implemented using modern micro-services architecture which is highly scalable and flexible as opposed to traditional monolithic architecture design.
- Micro-Services in a nutshell can be conceptualized as a new way to design and develop enterprise level applications, wherein the entire application is broken down as multiple services, which are independent of each other and of the language used to develop them.
- The web application follows this modern approach and the entire system is broken down into multiple independent services making the application loosely coupled with the client application, highly scalable and extensible.
- Some services are shown in the figure 2.1.1 which are exposed as RESTful resource.

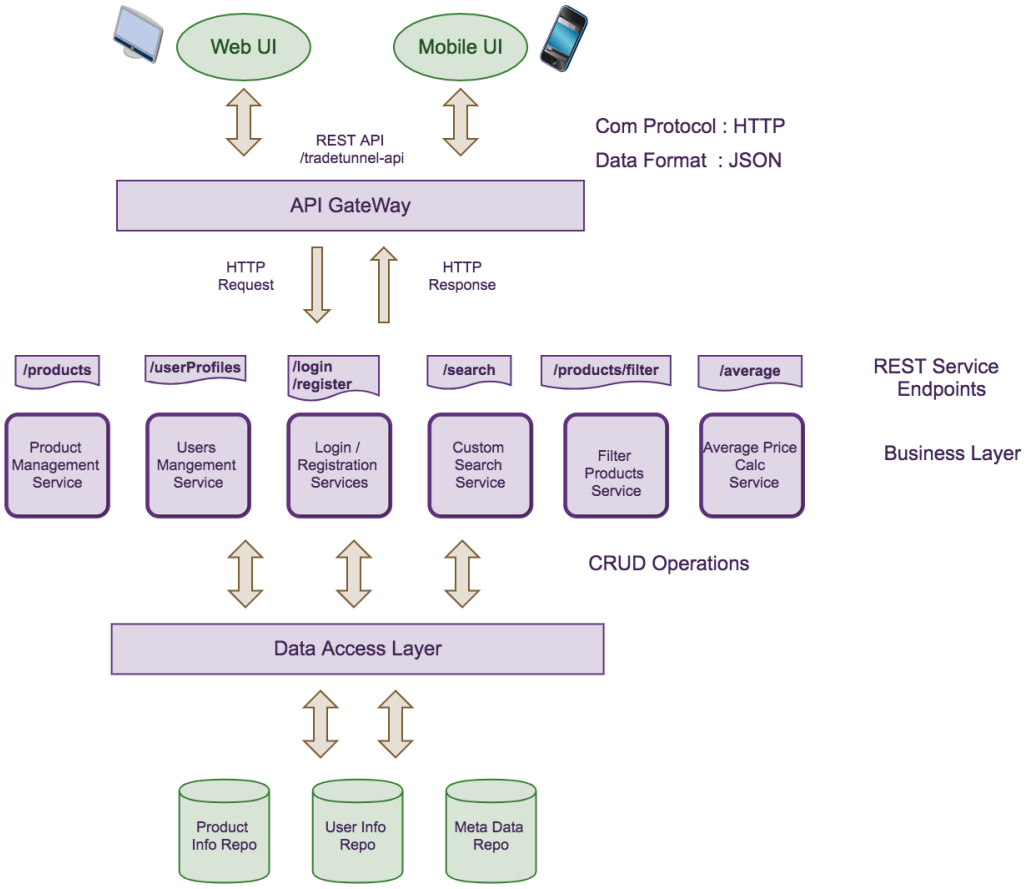


Figure 2.1.1

- As shown in the figure 2.1.1, the architecture can be broken down into three main layers the client application layer, the Business Layer and the Data Access Layer.
- **Client Application Layer** - The server side application can serve any client application e.g. Mobile device application, Web application etc. that supports and understands JSON data format with HTTP as an underlying communication protocol.
- **Business Layer** - The system is capable of servicing multiple clients at the same time.
  - Entire business logic is implemented at the business layer
  - The Business logic is exposed as multiple RESTful Web services which on receiving a request from the client, performs the necessary CRUD operations

  - Spring Data REST is used to develop and expose web services as resources using java. The resources are implemented in a hypermedia driven way.
  - Spring Data REST leverages hypermedia to allow clients to find functionality exposed by the repositories and integrates these resources into related hypermedia based functionality automatically.
- **Data Access Layer –** Spring Data JPA makes DAO implementation abstract and Repository interfaces are exposed which are used by the controllers on the business layer to perform the required CRUD operations on the database very easily and efficiently.

2.3. SYSTEM FLOW

- The client application consumes the web services, which are exposed by the server.
- Client application, the front end in our case initiates HTTP request targeting one of the exposed web-service end point. For e.g. */tradetunnel-api/login* service as shown in the figure 2.1.1
- The servlet container on the server is the gateway to the backend api which on receiving the client request, forwards the request to the indented web-service endpoint controller.
- The Business logic implemented in the controller is executed which mainly results in performing CRUD operations using repository interfaces exposed by the data access layer.
- Finally, a HTTP response is sent back to the client.

2.4. SERVER SIDE IMPLEMENTATION

- As mentioned earlier, server side is implemented using Spring Data REST which automatically exports repositories as web-service resources which are implemented in hypermedia driven way.
- A hypermedia driven web application provides information to navigate to all the REST interfaces of the exposed API dynamically by including the hypermedia links with the responses.
- Spring Data REST uses HATEOS (Hypermedia as the engine of Application State) architecture to implement hypermedia driven systems.

- Spring Data REST by default uses HAL (JSON Hypertext Application Language) to render responses for resource discovery. HAL defines the way in which the embedded resource and the links to other resources is returned in the response.
- The above can be understood more clearly with the following example of our own trade tunnel system.
- As shown in the figure 2.4.1 resource discovery starts at the context-path level (/tradetunnel-api) itself. A HTTP GET Request on the context root renders all the REST resources exposed by the Trade-Tunnel API as shown in the figure 2.4.1
- All the exposed resources are repositories which are explained in detail in section 2.6

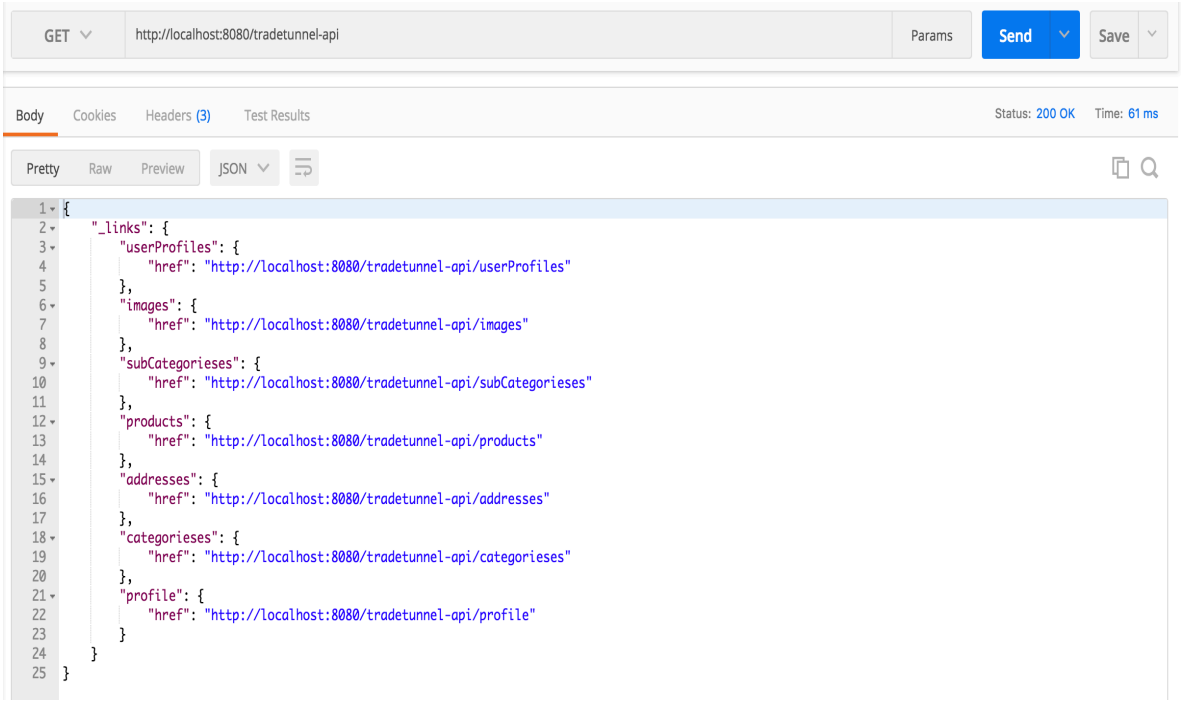


Figure 2.4.1

- Figure 2.4.2 shows the response of the HTTP GET request sent at userProfiles endpoint (the link to this endpoint was obtained from the response as shown in figure 2.4.1)

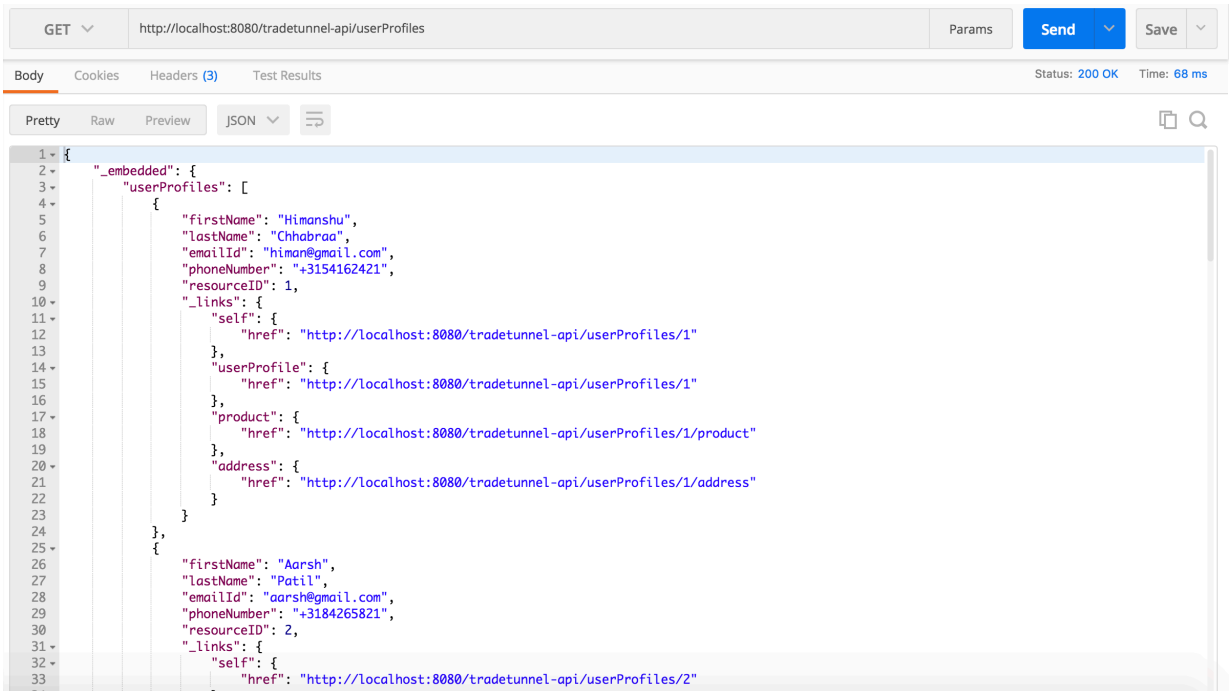


Figure 2.4.2

- The response is a Hypermedia response and the resource discovery is performed using HAL specifications, which means
  - The data present within the *_embedded* property is the response of the requested resource, which is a list of user profiles and the resource Ids can be considered as unique primary keys.
  - The links returned as a part of the response within the links property of each user profile in the list are the links to other resources specific to that user.
  - Which means, the client application on having the JSON response (HAL notation) can easily request for the users address and the products uploaded by the user on the website for sale.
  - This is a hypermedia driven site implementation where in the client receives links to all the exposed REST interfaces in a response making it much easier for client to make subsequent REST calls to the backend TradeTunnel API.
  - This approach eliminates the traditional WSDL distribution overhead and boiler plate coding.

- Figure 2.4.3 shows the response of the HTTP GET request sent at products endpoint (the link to this endpoint was obtained from the response as shown in figure 2.4.1)

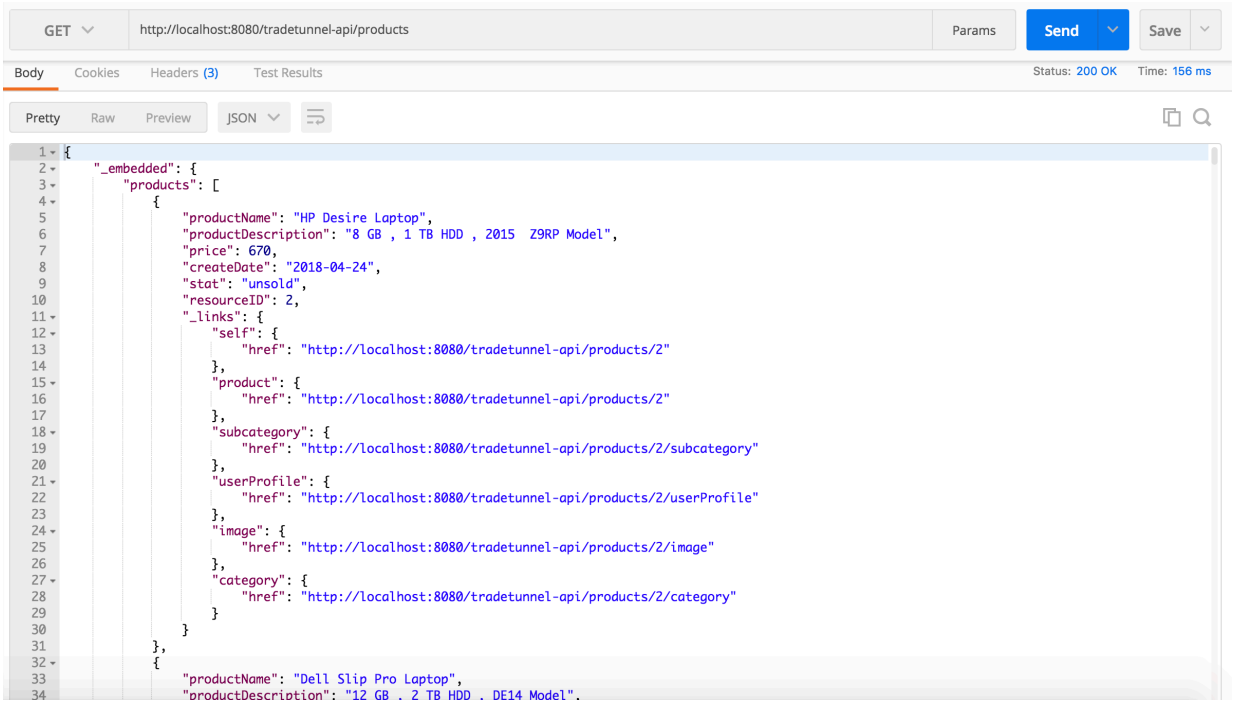


Figure 2.4.3

- Following with the explanation as mentioned earlier, following inference can be drawn from figure 2.4.3
  - The GET response returned a list of products in the system each following the HAL notation.
  - The data present within the *_embedded* property (product name, description, price etc.) is the data for the current resource (/products) requested,
  - The links present in the *_links* property gives access to other resources to perform operations on that product
  - The client application having the above response HAL object of a product can easily request the server to fetch information about the category and subcategory it belongs to, images of that product, the *userProfile* of the user who has uploaded the product for sale

- This is how multiple services of the application are broken down and are implemented as separate micro-services.
- HATEOS makes the process of publicizing the REST links of the API much easier as it exposes links to all the resources in every response.
- Server side of the Trade Tunnel web application has used the power of HATEOS concept to implement micro services using Spring Data REST.
- Additional services such as login, average price, search products, filter products are also exposed as distinct web-service end-points which are published to the consumer

2.5. CLIENT SIDE IMPLEMENTATION

- Following the architecture and designs so far it is clear that the front end and the backend of Trade tunnel as a web application are loosely coupled.
- This makes the application highly scalable and any kind of client can easily consume the services as exposed by the backend API
- The client side makes AJAX and synchronous and asynchronous web-service calls to the backend API to request for a service.
- The response is rendered by the frontend API and presented to the user.
- Following this approach, the client side requires no knowledge about the backend business logic as both are separate components of the software system.
- Client side makes use of *cookies* for Session management. To be more specific, on login, the backend API returns the logged in users HAL JSON response which is stored as a cookie on a local storage on the browser.
- Having said that, the client side now has access to all the resource links.

2.6. RELATIONAL DATABASE IMPLEMENTATION

- The application makes use of relational database system, implemented using MySQL server and MySQL workbench as the client
- The ER Diagram for the system (TradeTunnel Schema) is as shown in the figure 2.6.1
- The UserProfile table and the Address Table share one to many relationship
- The UserProfile table and the Product Table share one to many relationship
- The Product table and the Image Table share one to many relationship

- The Product table and the Categories table share many to one relationship
- The Product table and the SubCategories table share many to one relationship
- The Category table and the SubCategories table share one to many relationship

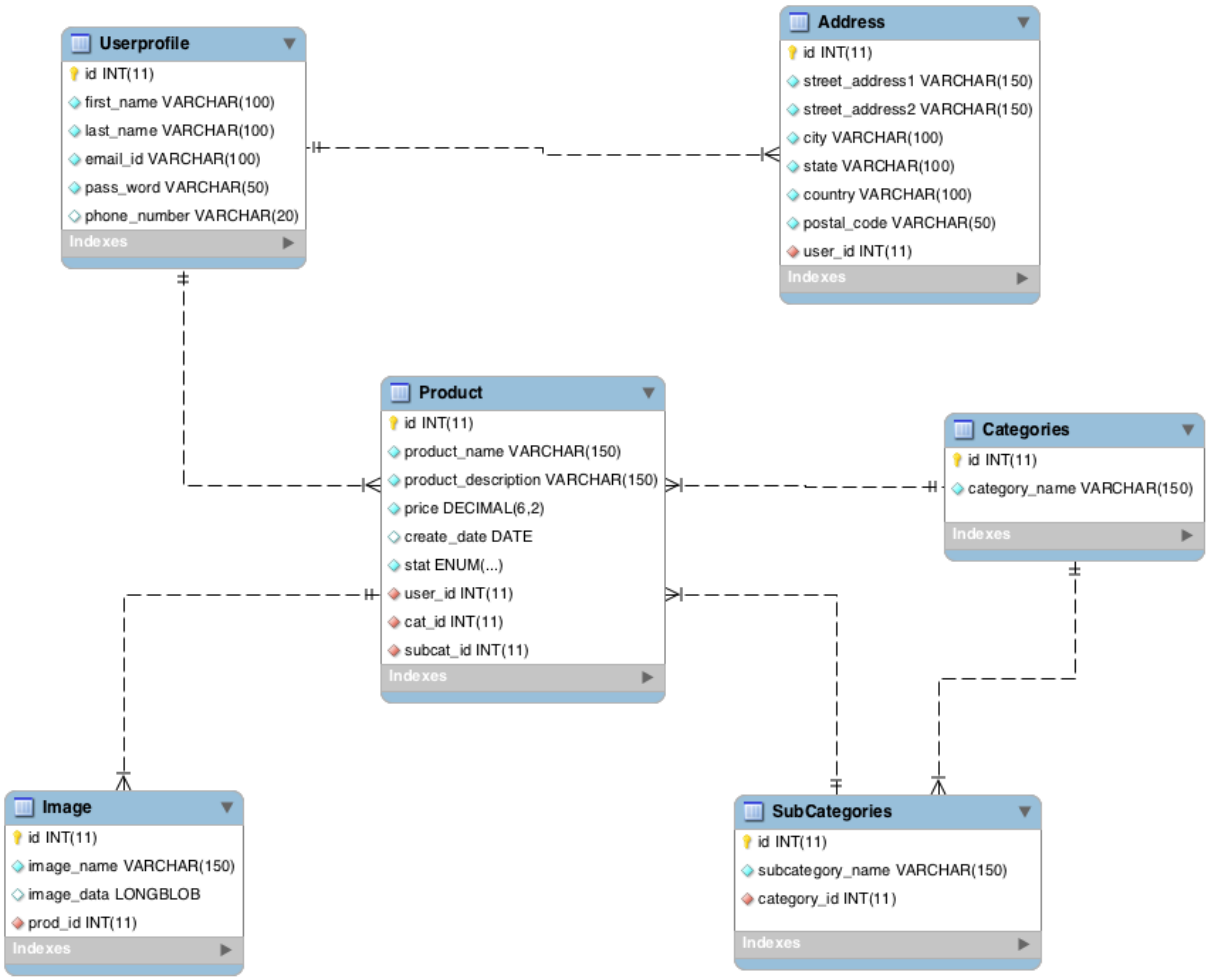


Figure 2.6.1

2.7. ADVANTAGES OF MICROSERVICES OVER MONOLITHIC ARCHITECTURE

- A Monolithic application is built as a single artifact, which means that the client interface, the server side code and the relational database system are tightly coupled.
- Any change in the front end or the backend system requires the entire logical unit (.ear or .war file) to be redeployed on the server.
- Moreover, this results in severe difficulties during application scalability and integration with external system.
- A monolithic application offers a limited reusability
- By Definition monolithic applications are implemented using a single development stack (e.g. JEE or .NET) which further adds dependencies and tight coupling to the software design.
- There are several advantages of using micro-services architecture over the traditional monolithic design, following is an exhaustive list
    - As Micro–Services encourage breaking the entire system into small independent distinct services, the development can be done independently and in parallel.
    - All the services exist as independent deployment artifacts and it can be scaled independent of other services.
    - Microservices are usually exposed using a standard protocol such as RESTful API.
    - Microservices can be easily consumed by the client application without direct coupling using shared libraries.
    - It gives freedom to the developers to choose the appropriate framework for the development of microservices
    - Maintenance of all the services can be carried out separately without affecting other services of the same API. This makes the application highly scalable and extensible.
- Trade-Tunnel adheres to the microservices standards and inhibits all the advantages as mentioned above, making it highly scalable, flexible and open to extension.
- The server side implementation can serve any kind of client requesting for services by consuming the exposed microservices rescources.

3. Development Stack

Frontend Technologies

Backend Technologies

Relational Database

Application Server

Version Control

Build Tool