Fulda Fall 2019
Global Distributed Software Development
HSF-Marketplace
WiSe 2019 Group: G3

Dan Jalba
Nikhil Yadav
Asmita Upreti
M.Usman Aslam
Kumkum Rajput
Jobayer Hossen Mojumder

November 21, 2019

1. Functional Requirement - Prioritized

User Registration:

1. **Signup:** A user can get himself registered by inputting his data for the website. He can add his/her credit card now or at a later stage but before checking out.
2. **Unique user ID:** If the user ID is already taken up, an alert is issued to enter another one. Enter the password and confirm it.
3. **Contact:** Prompt user to add his contact number, email, and postal address.
4. **Membership:** A user can select his/her membership type. A user can be a Gold Member or silver Member based upon his purchasing power.
5. **Save:** Update or database for the user record.

Login:

1. **Credentials:** A user can login to the website using valid Username/ID or password.
2. **Alert:** If the user entered wrong ID authentication failed error is generated.
3. **Reset:** If a user forgets his/her username or password he/she can rest it after getting reset link at registered email only.

Unregistered User:

1. Intro: An unregistered user can view our website. He can visit our home page and can search for products.
2. Wishlist: Unregistered user can add items to his/her wish list which he can access later on after registering as a registered user.

Buyer:

1. **Intro:** A Buyer is a registered user who can access our website. He/she can be either a Gold member, silver member or a normal member.
2. **Shopping cart:** A buyer will be able to search for products and similarly can add them to his/her wish list or shopping cart. He/she can also remove products from the shopping cart.
3. **Contact:** A buyer can also contact the seller for the detailed specification of the product, expected time of delivery.
4. **Authentication:** A buyer must authenticate his/her contact and information before proceeding with the buying of the items.
5. **Order Status:** A buyer can view his/her order status and can also track it. It also shows that whether or not he/she can cancel it or not.
6. **Cancellation:** In Case of cancellation of the order it shows the penalties and order cancellation. No order can be canceled after it has dispatched for delivery.

7. **Return:** A buyer can return the product if it is faulty and is not according to the specification in limited time.

Seller:

1. **Intro:** A seller is a registered user who can post products in our catalog. Only a registered member can be a seller. Sellers can also post ads for the sale of their products.
2. **Membership:** Only a Gold or Silver member can post ads or products for selling. A normal member has to be upgraded to become a seller from the buyer. The confirmation period is at least 1 week.
3. **Profile:** Seller's profile must be visible to buyers at the time of posting an ad or product. The buyer can easily contact the seller in case of return and non-delivery and any other just query.
4. **Dashboard:** Seller can post and manage ads and his/her products up for sale.
5. **Discounts:** Seller can offer discounts to customers based upon his own affordability. As our product will provide a platform to the seller to get market access to his/her products.

Super User "Admin":

1. **Intro:** An Admin is a superuser of our web system. He/she is the operator and guardian of the whole system. His account is special and is not visible at any level to other users.
2. **Admin panel:** A panel is available to Admin from where he/she can manage the whole web system. Admin can access the database and all records and can also grant and revoke permission to other users.
3. **Authentication:** Every registered user must be authenticated by Admin after validation. Admin can ban or deregister any user based on his activities.

Catalog:

1. **Intro:** Our products are visible to every user. The user can be a buyer, seller or unregistered user.
2. **Browse:** Users can browse our catalog at a very basic level. Every user has the same view of our catalog.
3. **Post Products:** Only a seller can post the product to our catalog which he is intended to sell others. For posting a product he must be a registered user with Gold or Silver Membership.

Shopping Cart:

1. **Intro:** A shopping cart is a personalized basket for buyer whether registered or not registered.

2. **Edit:** A buyer can edit shopping cart items before proceeding to checkouts.
3. **Shared cart:** Every user can share his/her shopping cart inside our web system. They can do so by sharing the links and articles numbers with each other without using any third-party platform.

Order Tracking:

1. **Intro:** A buyer can track his order after the order is confirmed. A confirmation number is assigned to the order which is used to check the order status.
2. **Cancellation:** A buyer can cancel the order after its confirmation but the order must not have to be dispatched. Otherwise, he is not eligible to cancel it.
3. **Return order:** In case of wrong specifications and damaged delivery the order can be returned back in a fixed amount of time and we provide the basic level necessary support.

User Friendly:
The system should be user-friendly and provide an interactive user interface. The operational procedures should be robust yet simple enough to be understandable to every type of user.

It should also be easily maintainable and accessible by the administrator. Admin can easily perform his tasks through the dash panel.

Performance:
The system should perform in different environments. It should handle all the traffic and should have enough space for all the products.

Backups should be updated frequently and regular updates for new technologies and components should be performed more often.

Security:
The system should be free of any loopholes. The system should be protected from external threats and breakdowns.

Customer data is of sensitive nature so complex algorithms should be implemented to safeguard their transactions.

Prioritization

Functional Requirement	**High Priority (Must have)**	**Medium Priority (Should have)**	**Low Priority (Could have)**
User Registration	● Signup ● Unique ID ● Contact ● Save	● Membership	
Login	● Credentials ● Alert	● Reset	
Unregistered User	● Wishlist		
Buyer	● Shopping Cart ● Contact ● Order Status ● Cancellation	● Authentication	● Return
Seller	● Profile ● Dashboard	● Discounts	
Admin	● Admin Panel	● Authentication	
Catalog	● Browse ● Post Products		
Shopping Cart	● Edit		● Shared Cart
Order Tracking	● Cancellation		● Return Order
Other features	● User-Friendly ● Performance ● Security		

3. High-level Architecture, Database Organization

DB organization: The main database schema/organization (high level)

The database consists of the following tables-

1. **superuser**

   Admin information is added to the database with a unique ID. Along with other necessary columns, this table has a Role column that identifies a particular admin user.
   superuser_id: Stores the unique Id of the superuser
   firstname: The first name of the user
   lastname: The last name of the user
   password: Stores the encrypted password. In PHP, we use the password_hash() function, which uses Bcrypt(a key-strengthening hash algorithm) by default.
   email: Email ID of the user
   role: Gives a role to the superuser. It can be either an admin or a test user.
   privilege: Gives a set of privileges assigned to a role.

Field Name	**Datatype**
superuser_id	INT(10), Not Null, Unique
firstname	VARCHAR(30), Not Null
lastname	VARCHAR(30), Not Null
password	VARCHAR(70), Not Null
email	VARCHAR(50), Not Null, Unique
role	VARCHAR(20), Default=test
privilege	VARCHAR(20), Default=read/write

2. **users**

   The user's information is added to the database with a unique ID. This has columns as-
   user_id: Stores the unique Id of the user
   firstname: First name of the user
   lastname: Last name of the user

password: Stores the encrypted password. In PHP, we use the password_hash() function, which uses Bcrypt(a key-strengthening hash algorithm) by default.
initial_email: Non verified email of the user. This is before the user's email id is not verified.
city: City of the user.
state: State of the user.
zip: Postal Code of the user.
email: Verified email of the user.
verification_code: Generated code sent on the email to get the email verified.
user_ip: IP Address of the user
phone: A 10 digit phone no. of the user
country_code: +49 for Germany, and +1 for US, to add before the phone no.
country: Country of the user.

Field Name	**Datatype**
user_id	INT(10). Primary key in this table. Not Null, Unique. Acts as foreign key in tables- wishlist, order.
firstname	VARCHAR(30), Not Null, Unique
lastname	VARCHAR(30), Not Null, Unique
password	VARCHAR(70), Not Null
inital_email	VARCHAR(50), Not Null, Unique
city	VARCHAR(30), Not Null
state	VARCHAR(30), Not Null
zip	VARCHAR(10), Not Null
email	VARCHAR(50), Not Null, Unique
verification_code	INT(10), Not Null, Unique
phone	VARCHAR(10), Not Null
country_code	VARCHAR(3), Not Null
country	VARCHAR(30), Not Null
is_buyer	BOOLEAN, Not Null

is_seller	BOOLEAN, Not Null

3. **product**

A product defines an item to buy/sell. Complete product information is stored in this table.
product_id: Stores the unique Id of the product table.
product_sku: Stores the unique Id of the product itself.
name: Name of the product.
cost: The cost of the product.
category_id: The Id of the category under which the product falls into.
category: The category name of the product.
image: The link to a full-size image.
thumbnail: The link to the thumbnail of the image.
description: Description of the product.
stock: Number of the item left in the inventory.
status:

Field Name	**Datatype**
product_id	INT(10). Not Null, Unique. Primary key in this table.
product_sku	VARCHAR(30), Not Null, Unique
name	VARCHAR(30), Not Null
cost	DOUBLE(10), Not Null
category_id	VARCHAR(30), Not Null
category	VARCHAR(30), Not Null
image	VARCHAR(100), Not Null
thumbnail	VARCHAR(100), Not Null
description	VARCHAR(300), Not Null
stock	INT(5), Not Null
status	ENUM (‘pending’, ‘approved’, ‘denied’, ‘sold’)

4. **order**

The ordered products, its status, and delivery information are stored in this table.
order_id: Stores the unique Id of the order table
order_no: Stores the unique order no.
amount: Total amount paid by the buyer.
buyer_id: Stores the user Id of the buyer.
seller_id: Stores the user Id of the seller.
timestamp: Date and time of the order placed.
tracking_no: A unique id to track the product.

Field Name	**Datatype**
order_id	INT(10). Primary key in this table, Not Null, Unique
order_no	VARCHAR(10), Not Null, Unique
product_id	INT(10), Not Null, Unique
amount	DOUBLE(10), Not Null, Unique
buyer_id	same as user_id in user table, Not Null.
seller_id	same as user_id in user table.Not Null.
timestamp	TIMESTAMP, Not Null
tracking_no	VARCHAR(10), Not Null, Unique.

5. **wishlist**

This contains a user's wishlist.
user_id: Stores the unique Id of the wishlist table
ipaddress: Stores the IP address of the user
product_id: Stores the product id of the products added in the list.

Field Name	**Datatype**
user_id	INT(10). Primary key in this table, Not Null, Unique
ipaddress	VARCHAR(15), Not Null

product_id	INT(10), Not Null, Unique

Media storage:

There is no direct media storage. Only the links to the images and thumbnails. There is no other special data format requirements.

Search/filter architecture and implementation:

Algorithm for search- Since each table has an index, the values are stored in B-Tree. Internal sorting and searching algorithms are applied by PHP.
Organization of search items- Searched items will be shown in a tabular format.
Database terms that can be searched are-

product_id
product_sku
name
category

Search on any other data will not give any result.
Coding and organization in the database- We use the search queries using wildcards.

APIs (high level): Major APIs that we will create other than standard ones are-

SearchAPI : Sends items to frontend based on a search query provided in the request

ProductAPI: Sends all items of a certain user to the frontend. Differs between registered users and unregistered users. Just sends approved items if the requested user is not the same as the requesting registered user, otherwise, it will additionally send items with a status of pending, denied, and sold. It does not include description.

MessageAPI: Sends all messages of a certain user to the frontend

WishlistAPI: Sends all items of the wishlist of a certain user to the frontend

AddWishlistItemAPI: Creates a new wishlist item in the wishlist of a certain user

DeleteWishlistItemAPI: Deletes an item in the wishlist of a certain user

CategoryAPI: Sends all items of a certain category to frontend

PendingItemsAPI: Sends a list of items that are pending approval. The backend has to check if requesting user is a moderator

PostItemAPI: Post item of a user. Includes approval by a moderator.

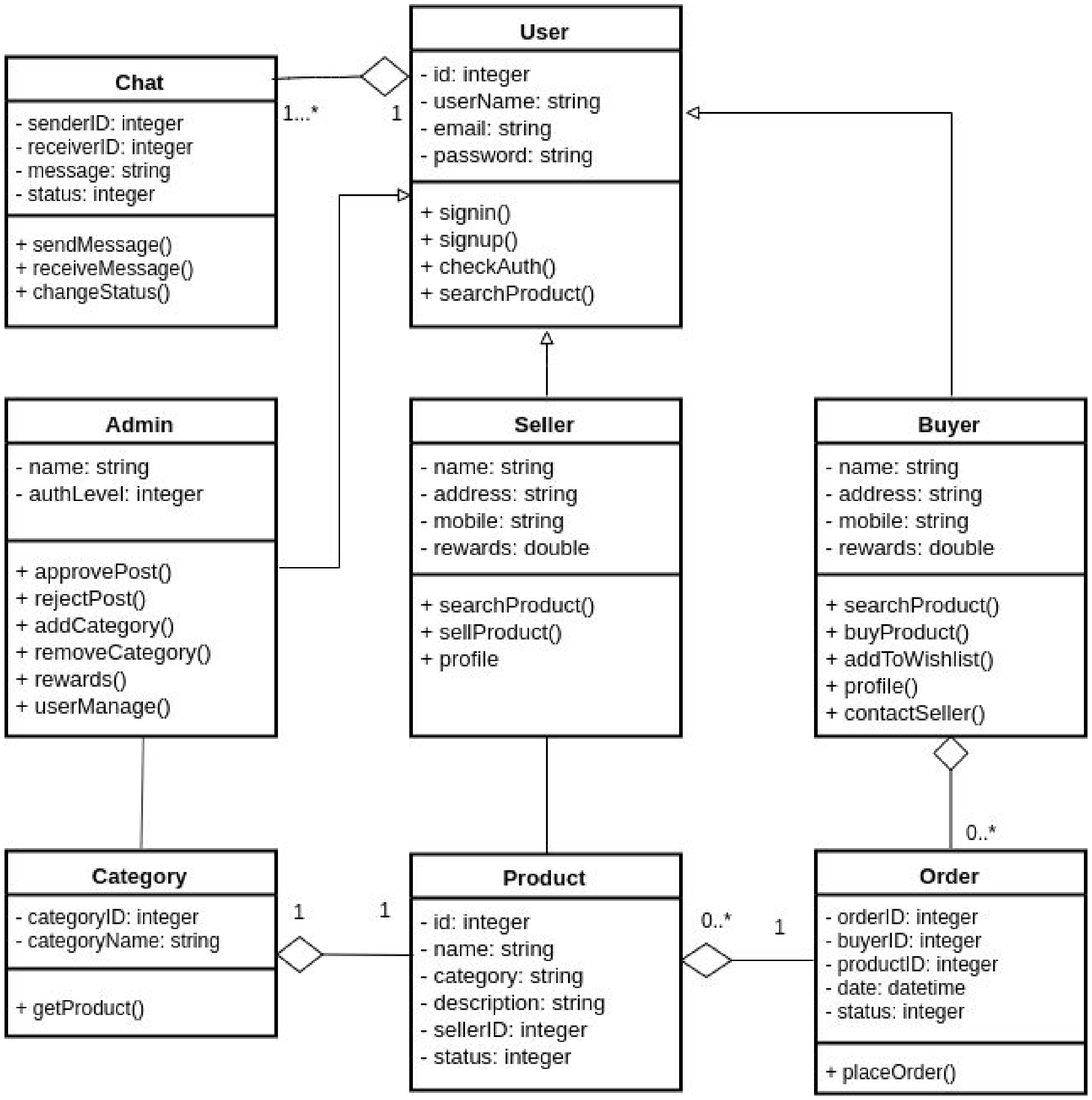
DeleteItemAPI: Delete item of a certain user

UpdateItemAPI: Update item of a certain user (description, price, title, etc.)

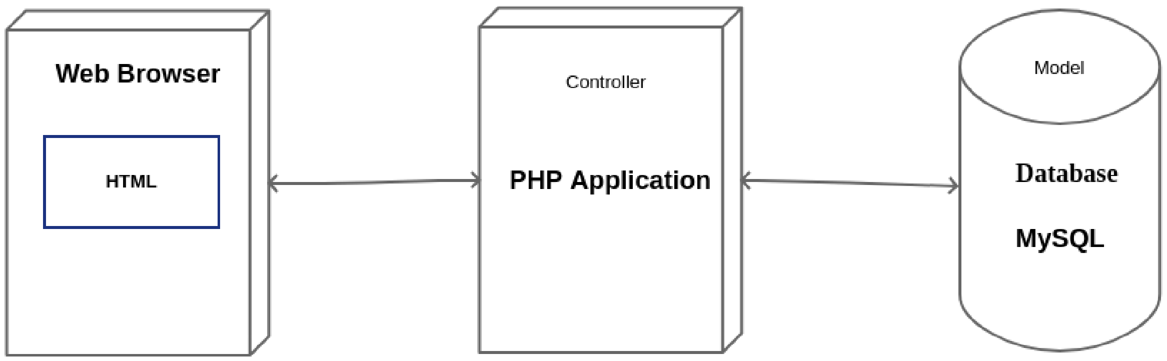
AddUserAPI: Creates a new user based on the information the unregistered user made to get a registered user

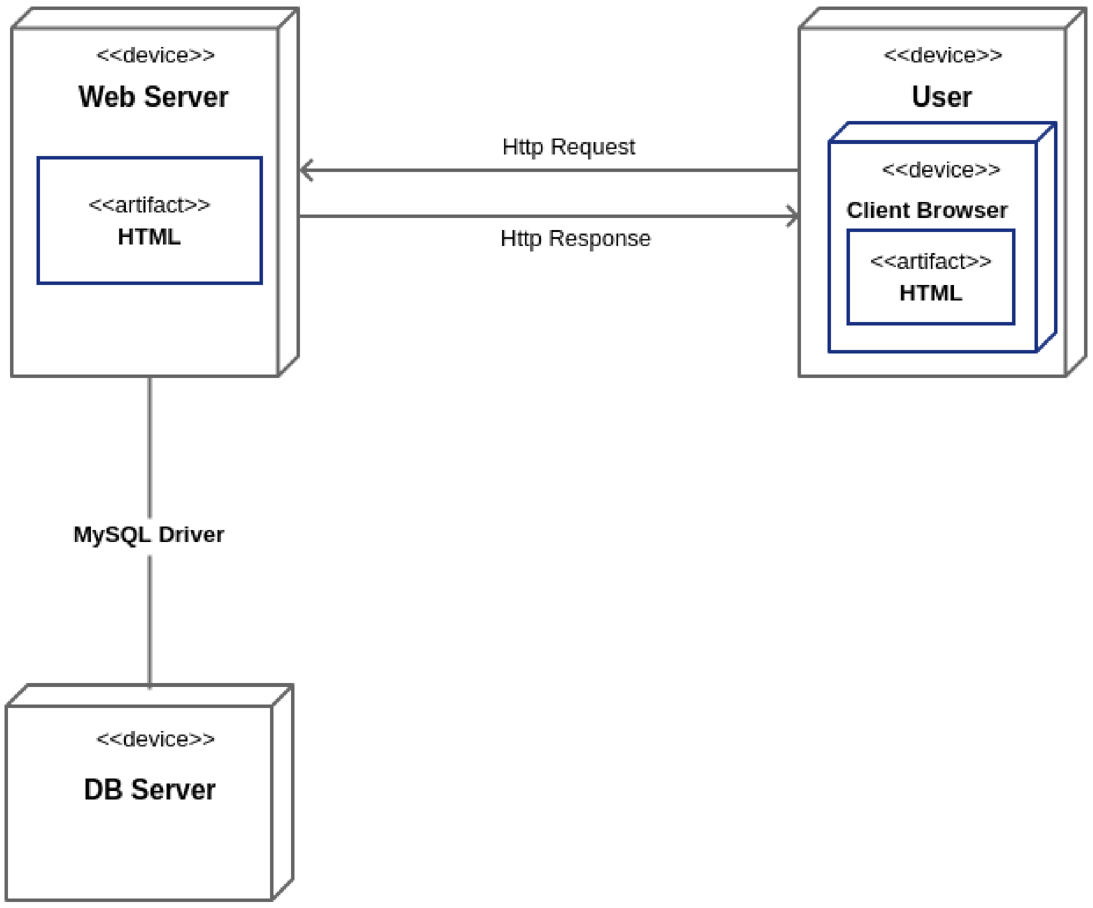
4. High-Level UML Diagrams

Class Diagram:



Component Diagram:





5. Risk Identification

1. **Skill risk:**
   Lack of skill to work on an existing part of the program written by another team member
   The team members will need to co-ordinate and explain the parts of the program and make it more understandable

   Lack of Skill to identify and rectify errors
   The member of the team struggling in this process have to either learn to test the program or ask other teammates for help and explanations

   Lack of Skills to implement a particular feature
   The member of the team struggling to implement a feature will need to go through tutorials to research the logic required to implement it.

2. **Schedule Risk:**
   Risk of not being able to deliver the product at a specified time. The Team members will hold a meeting every week in order to know about the progress of the project and to make sure everything is going on as planned. The team will focus on delivering the functionality of priority one feature and after completion on that will move to priority two features. Thus, even if the team won't be able to implement the priority feature will have all the functionality that the team has promised to deliver.

3. **Technical Risk:**
   Risk of members not being able to adapt to the technology that the group has decided. The members with prior experience in the technology will help other members facing the issues. The team consists of few members who have prior experience in the technology that the group has been using. Thus, in case of the occurrence of this risk, the member-facing difficulties will receive help and guidance from experienced members.

4. **Teamwork Risk:**
   Risk of members falling ill or they have to go back to their home country. In such case, the work of the absent member would be equally divided among the group members so that the absence of a member won't affect the development process.

5. **Legal/Content Risk:**

Risk of using unlicensed elements in the project like external libraries, images, or allowing people to buy or sell stuff that doesn’t abide by the laws placed in order by the government. This risk can be mitigated by keeping track of all the resources being used in the project and administering the content which will be sold bought and viewed on the website. The team will make sure to use only licensed libraries and media content.

6. Project Management

Managing team members took several steps before we start to actually work on the task. In the first step, we had a group meeting where we discuss things we have to do, discuss the notes we have done during feedback, and discuss requirements. After that, we prioritize all the tasks, which are the most important, and which are the less important. All this happens during our group meeting, and only on paper. After that, we discuss who can do what, based on his or her personal skills, knowledge and how fast can it be learned in case we need to learn something new. When we are done with discussions, our team leader gives tasks to every independent member, what to focus the most on that specific task and a deadline. Most of the time deadlines are the same for everybody, but there are some special cases, depending on tasks, skills, and workload from other courses. We use WhatUp to communicate remotely, it is simple and available for everyone, Google drive for documents, and GitHub for work. This is from our perspective a very efficient and clean way to work on a common project or document, no time is wasted to pass the document or project folder around. All tasks are registered and tracked by our team leader in Trello, an online management tool, where you can easily keep track of what you have done so far, what is going on, and what has to be done. When we are working on documentation, everybody is doing his part on his computer, in a document, and then, use Google doc, to merge with the main document, and we respect the order in which document is edited, person who did the first part, adds his part first to the main document and so on. When it comes to programming part, in order not to mess things around, break our server, and be time-efficient, we work in a very close collaboration, especially front end with back end, and work on the same element of the project, in parallel, merge everything in the end and test to make sure everything works fine, and only after that, we mark it as done, and proceed to the next part of the project.