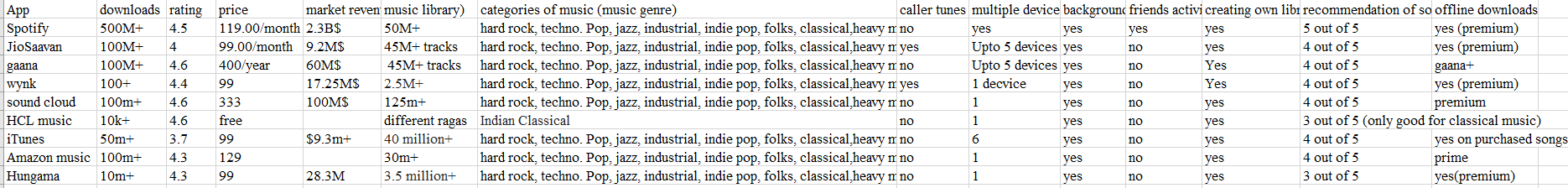
Practical 1

Study **SDLC** cycle and explore any android music app and do the market survey with respect to

Functionality and feature.



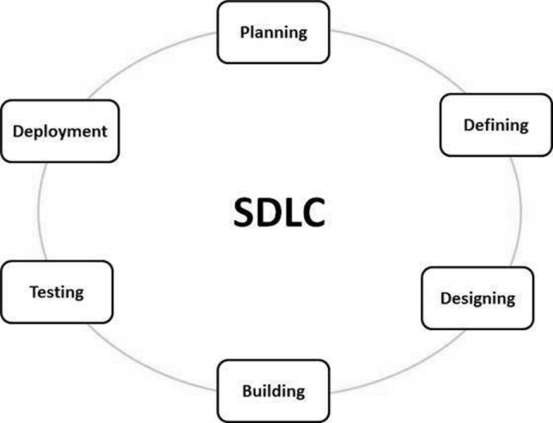
Here we have done market survey of 9 different music app on basis of following feature and functionality

1. Downloads
2. Ratings on downloading platform (Google play store)
3. Price of subscription
4. Market revenue
5. Music library
6. Category of music
7. Caller tune facility
8. Multiple device supports
9. Background working facility
10. Friends Activity
11. Creating own library
12. Recommendation of song
13. Offline download

The systems development life cycle (SDLC) is a process for planning, creating, testing, and deploying an [information system](https://en.wikipedia.org/wiki/Information_system). The systems development life cycle concept applies to a range of hardware and software configurations, as a system can be composed of hardware only, software only, or a combination of both.

There are usually six stages in this cycle: requirement analysis, design, development & testing, implementation, documentation, and evaluation.

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.

****

**Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, and various stake-holders of the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

**Stage 2: Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through SRS. Software Requirement Specification document which consists of all the product requirements to be designed and developed during the project life cycle.

**Stage 3: Designing the product architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a Design Document Specification.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules. The internal design of all the modules of the proposed architecture should be clearly defined with minimum details in DDS.

**Stage 4: Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc. are used to generate the code.

**Stage 5: Testing the Product**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

**Stage 6: Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations’ business strategy. The product may first be released in a limited segment and tested in the real business environment

Practical 2

Select user and domain for your project, write down your abstract about project and functionality.

Users: Admin, Buyer, Seller

Abstract: Our project is of an Auction website, where user can sell and buy products in given deadlines. Seller can put base price for the item he/she wants to sell. Plus additional detail about the deadline should be given by seller, in which buyer could buy the item.

1. **Tasks perform on website.**

Tasks:

1. Sign-up

* Add email address and password
* Add delivery address

1. Choose option

* Buyer option
* Seller option

For Sellers:

1. Add items

* Add item details
* Add base price
* Suggest tags for item
* Add time limit

For Buyers:

1. Buy items

* Category search for buying items
* Incremental bid on chosen item
* Payment of purchased item

For website manager:

1) Manage website

* Give continuous updates on website
* Solve disputes

Practical 3

Explore software process model compare these models and which is suitable for you justify



**When To Use?**

**Waterfall:**

In principle, the waterfall model should only be applied when requirements are well understood and unlikely to change radically during development as this model has a relatively rigid structure which makes it relatively hard to accommodate change when the process in underway.

**Incremental:**

This model can be used when the requirements of the complete system are clearly defined and understood. Major requirements must be defined; however, some details can evolve with time. There is a need to get a product to the market early. A new technology is being used. Resources with needed skill set are not available.There are some high risk features and goals.

**RAD:**

RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.It should be used if there’s high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

**Spiral:**

When costs and risk evaluation is important for medium to high-risk projects.Long-term project commitment unwise because of potential changes to economic prioritiesUsers are unsure of their needs .Requirements are complex.

**Prototype:**

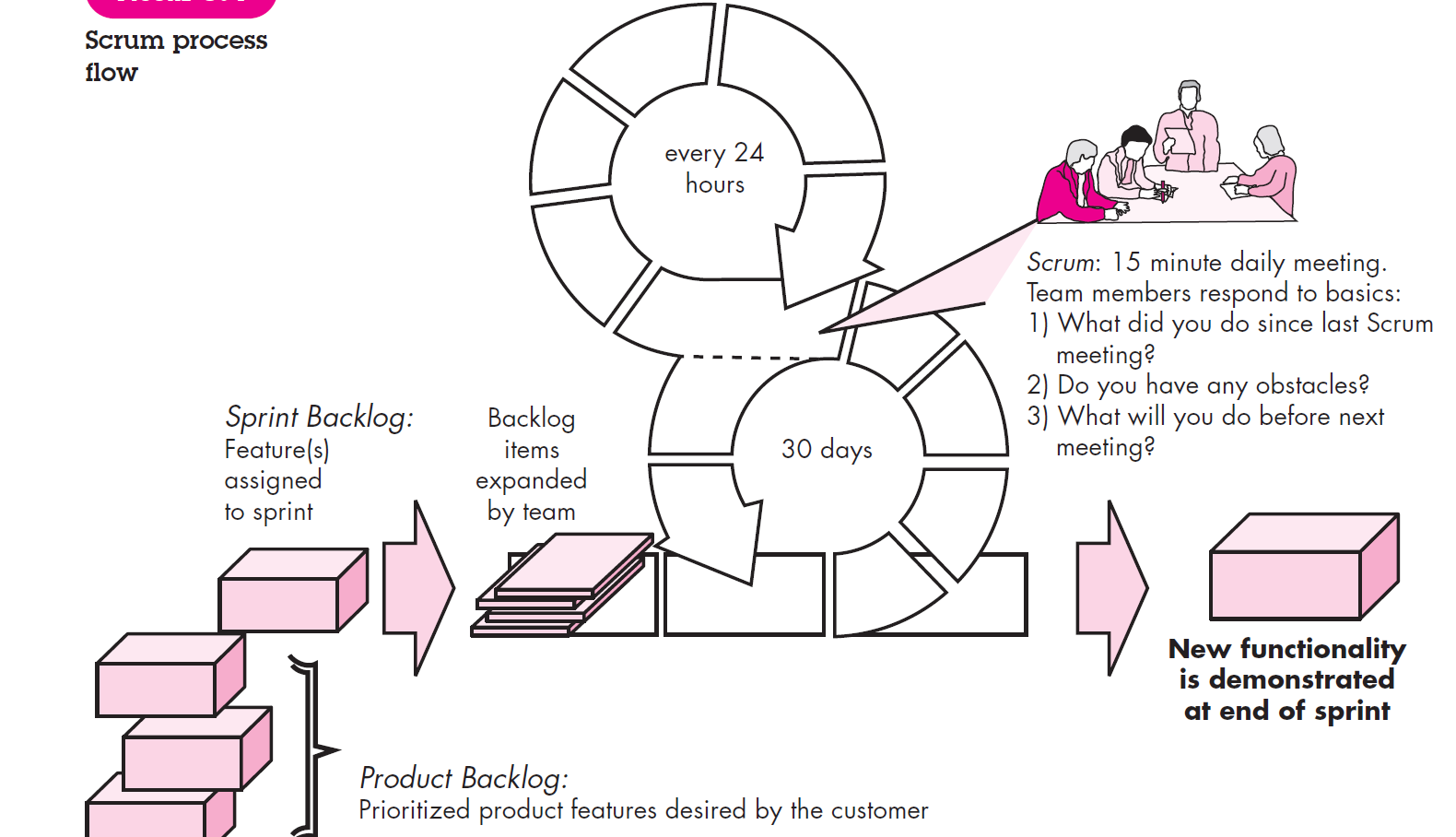
Prototype model should be used when the desired system needs to have a lot of interaction with the end users.

Which is suitable for you?

Waterfall model is suitable for us, because we don’t have teams to work parallelly work for requirement gathering, design, implementation etc. So, we have to first complete the requirement gathering and have to start work on designing and then implementation. We don’t have backward option, we have to only look for completing project on basis of previously gathered information in given deadline. That’s why Waterfall model is suitable for us.

SCRUM:

Scrum is an agile software development method that was conceived by Jeff Sutherland and his development team in the early 1990s. Scrum principles are consistent with the agile manifesto and are used to guide development activities within a process that incorporates the following framework activities: requirements, analysis, design, evolution, and delivery. Within each framework activity, work tasks occur within a process pattern called a sprint*.* The work conducted within a sprint (the number of sprints required for each framework activity will vary depending on product complexity and size) is adapted to the problem at hand and is defined and often modified in real time by the Scrum team.



*Backlog*—a prioritized list of project requirements or features that provide business value for the customer. Items can be added to the backlog at any. The product manager assesses the backlog and updates priorities as required.

*Sprints*—consist of work units that are required to achieve a requirement defined in the backlog that must be fit into a predefined time-box14 (typically 30 days).

*Scrum meetings*—are short (typically 15 minutes) meetings held daily by the Scrum team. Three key questions are asked and answered by all team members:

• What did you do since the last team meeting?

• What obstacles are you encountering?

• What do you plan to accomplish by the next team meeting?

A team leader, called a Scrum master*,* leads the meeting and assesses the responses from each person. The Scrum meeting helps the team to uncover potential problems as early as possible.

Practical 4

**Requirement gathering: Technical & Non-technical**

**Functional requirements** define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements are describing the behaviour of the system as it correlates to the system's functionality.

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

**Non-functional Requirements** can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviours of the system.  
Non-functional requirements are divided into two main categories:

* Execution qualities: like security and usability, which are observable at run time.
* Evolution qualities: like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioural requirements.

Technical requirements:

* Product upload
* Showcase of products
* Sorting & searching facilities
* Efficient timer for auction
* Payment facilities
* Showing latest bid with no database error
* Reminder of auction time

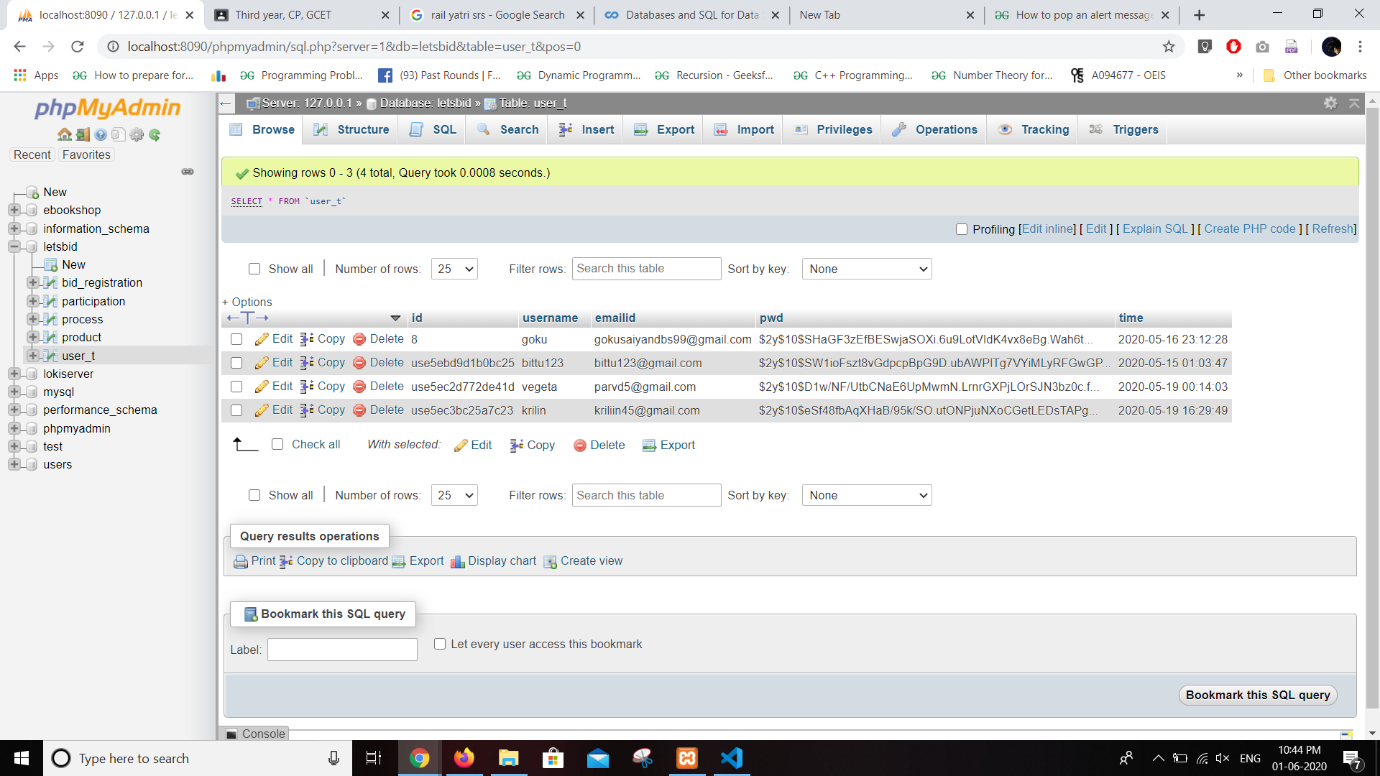
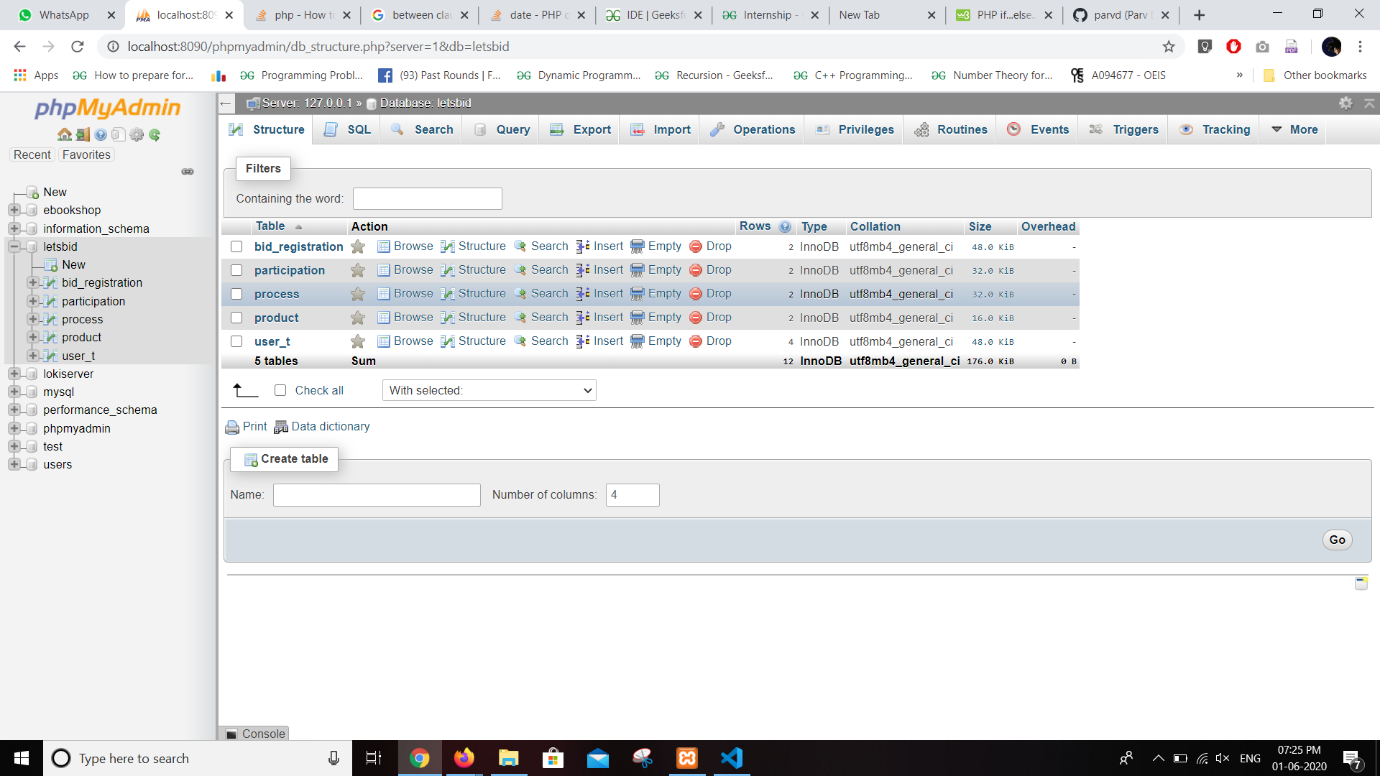
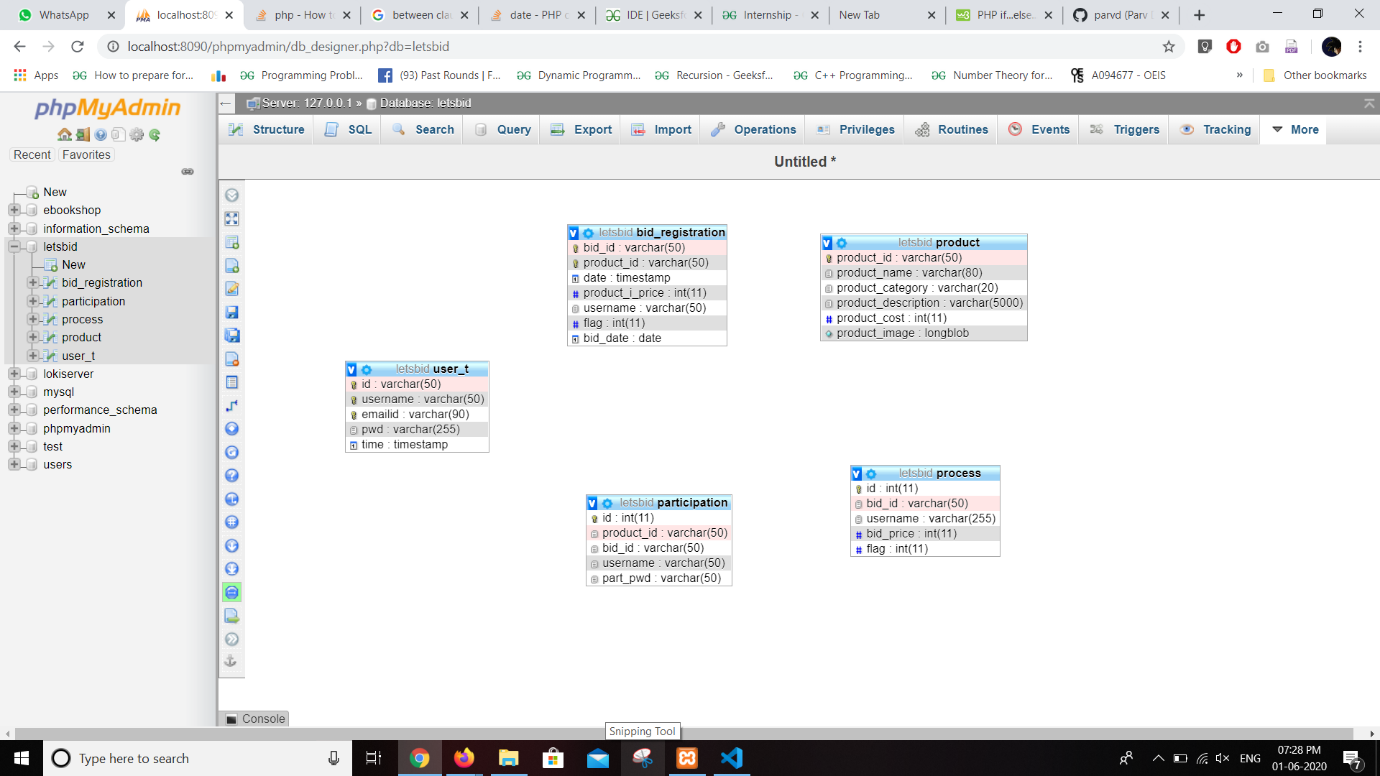
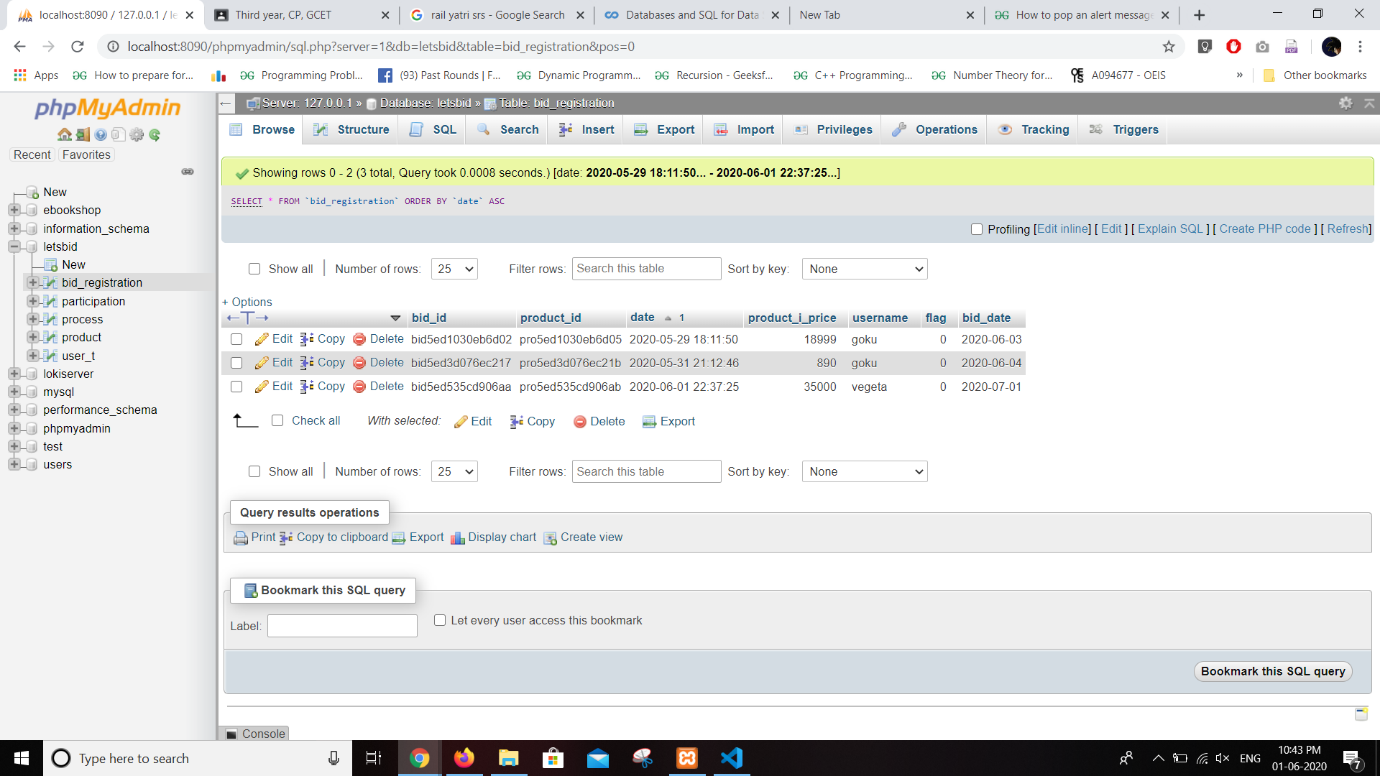
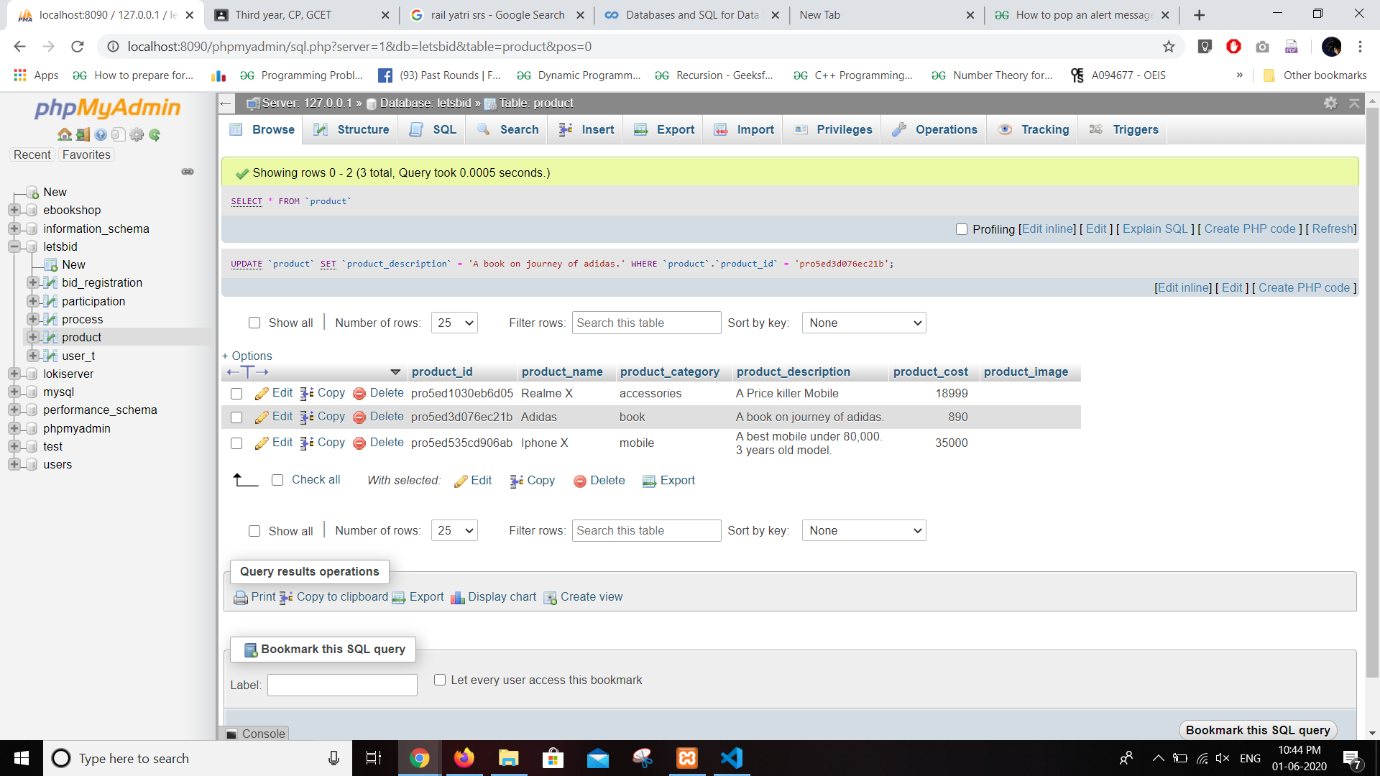
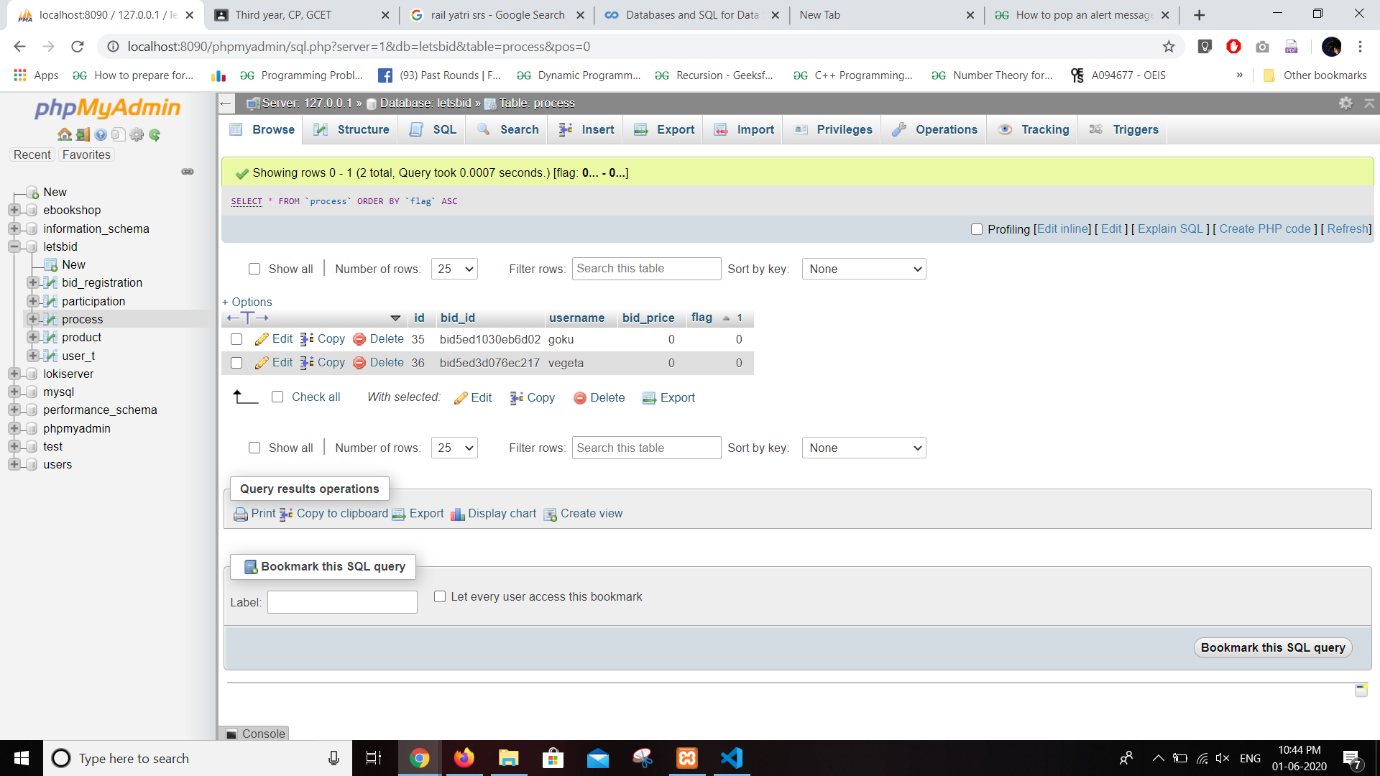
Non- technical requirements:

* 24\*7 bidding facilities
* Delivery facility of purchased product
* Authenticity of uploaded item

Practical: 4

Design & Implementation

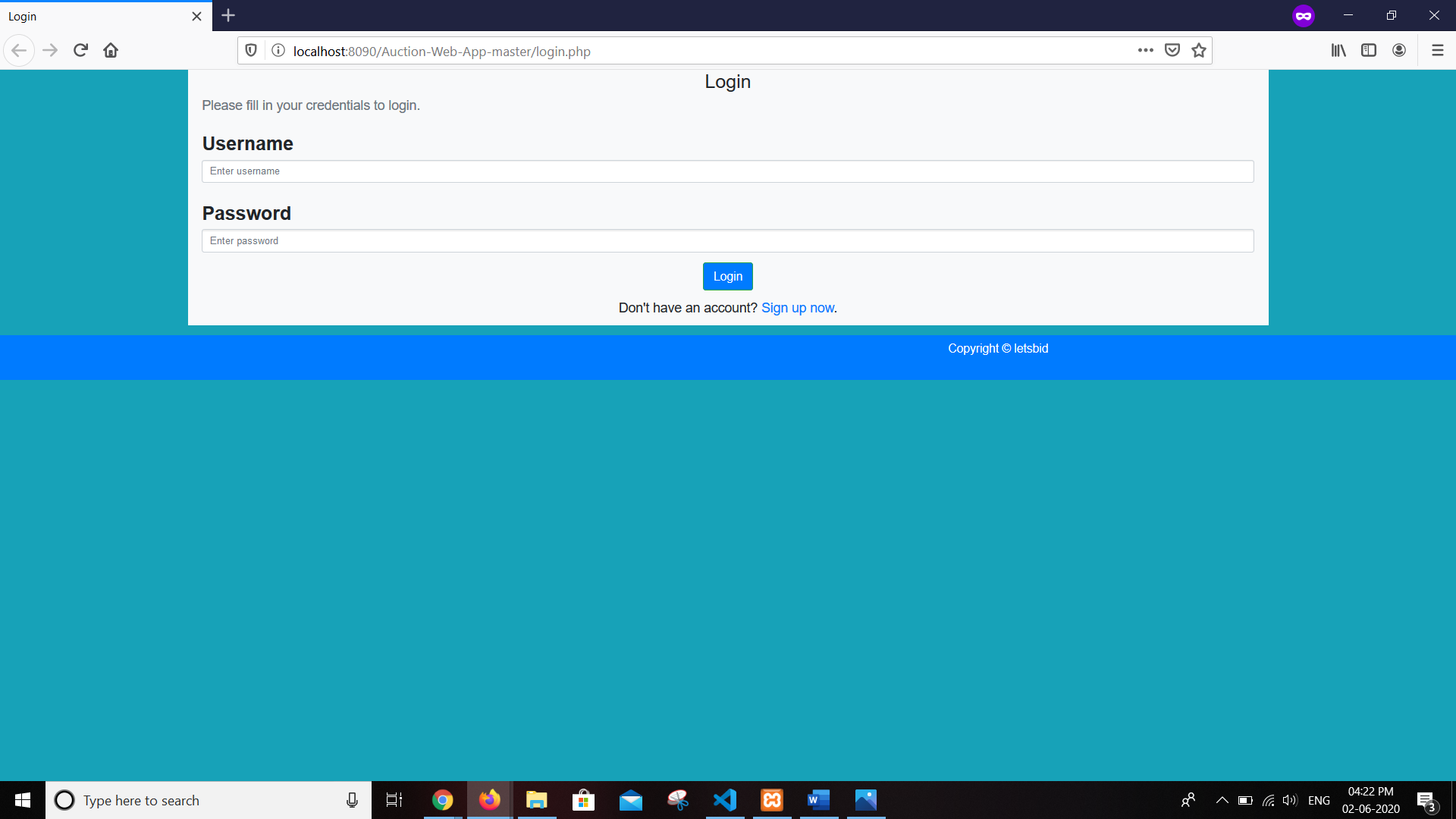
Database schemas:

s

Login / Signup:

File: login.php , signup.php

Signup requires username, email-id, password. Login using the same credentials as od signup.

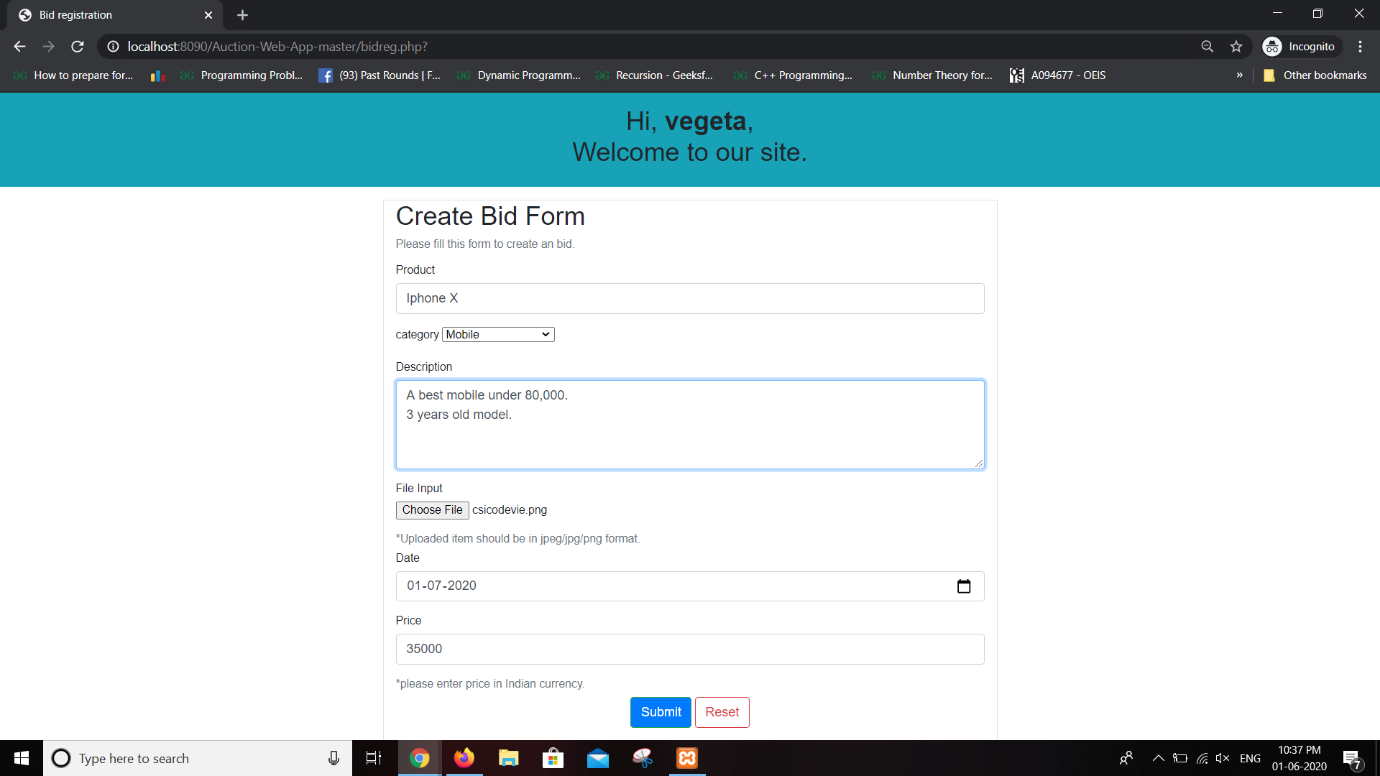


Bid registration:

File: bidreg.php

Bid registration includes:

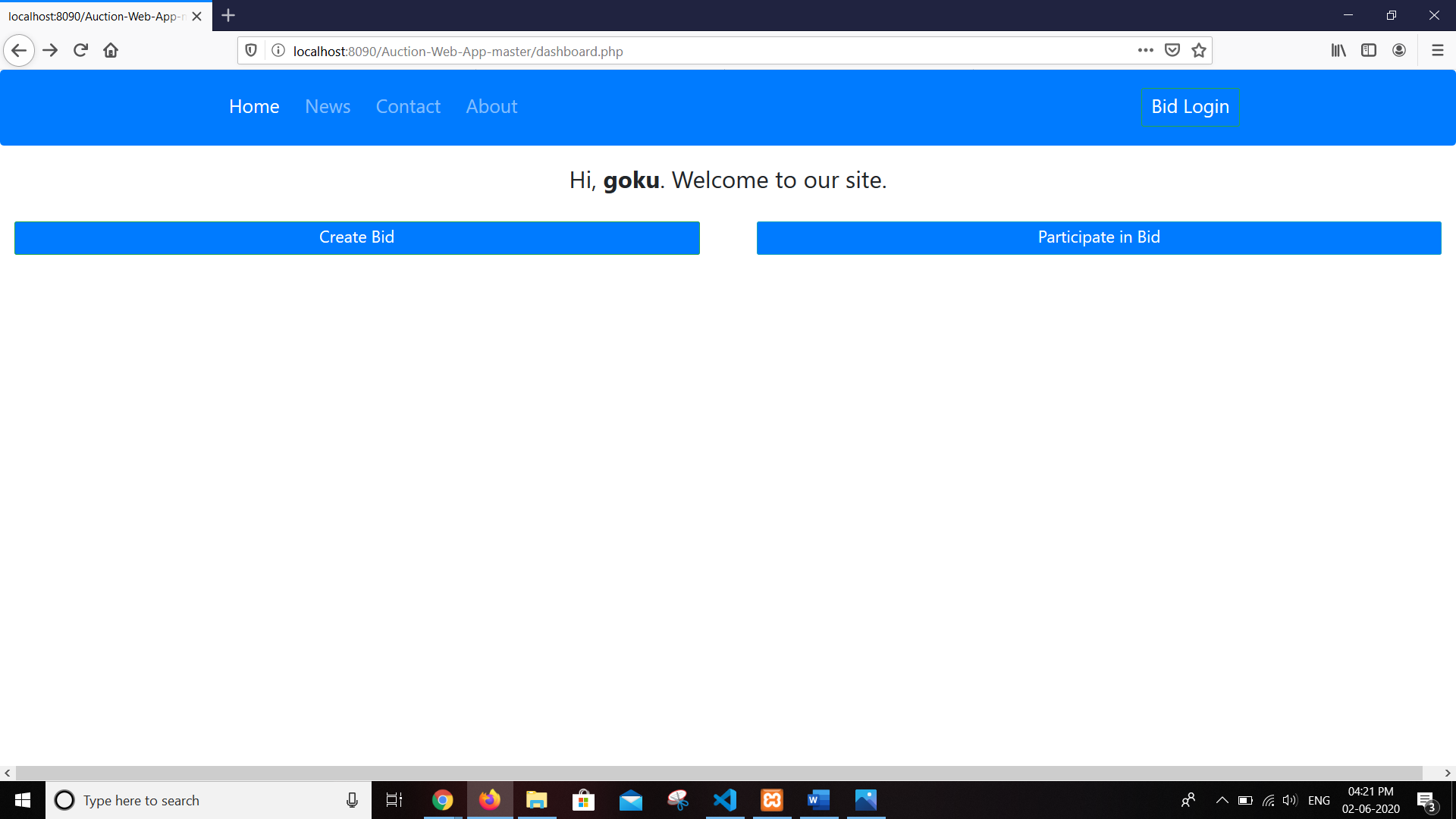
* Product name
* Category of product
* Description
* Photo of product
* Date of the auction
* Price (INR)



Dashboard:

File: dashboard.php

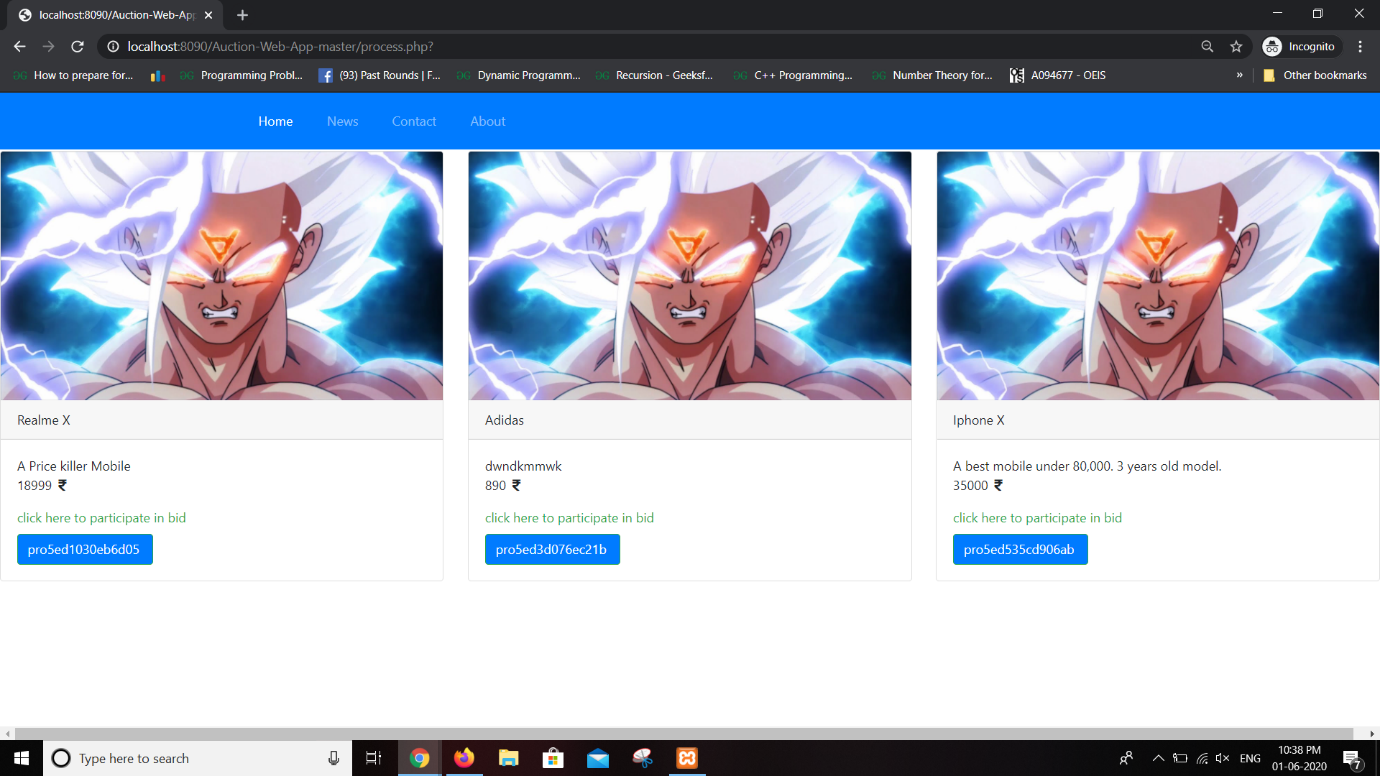
Dashboard includes hyperlinks for registering bid and participate in bid.



Participate in bid:

File: process.php

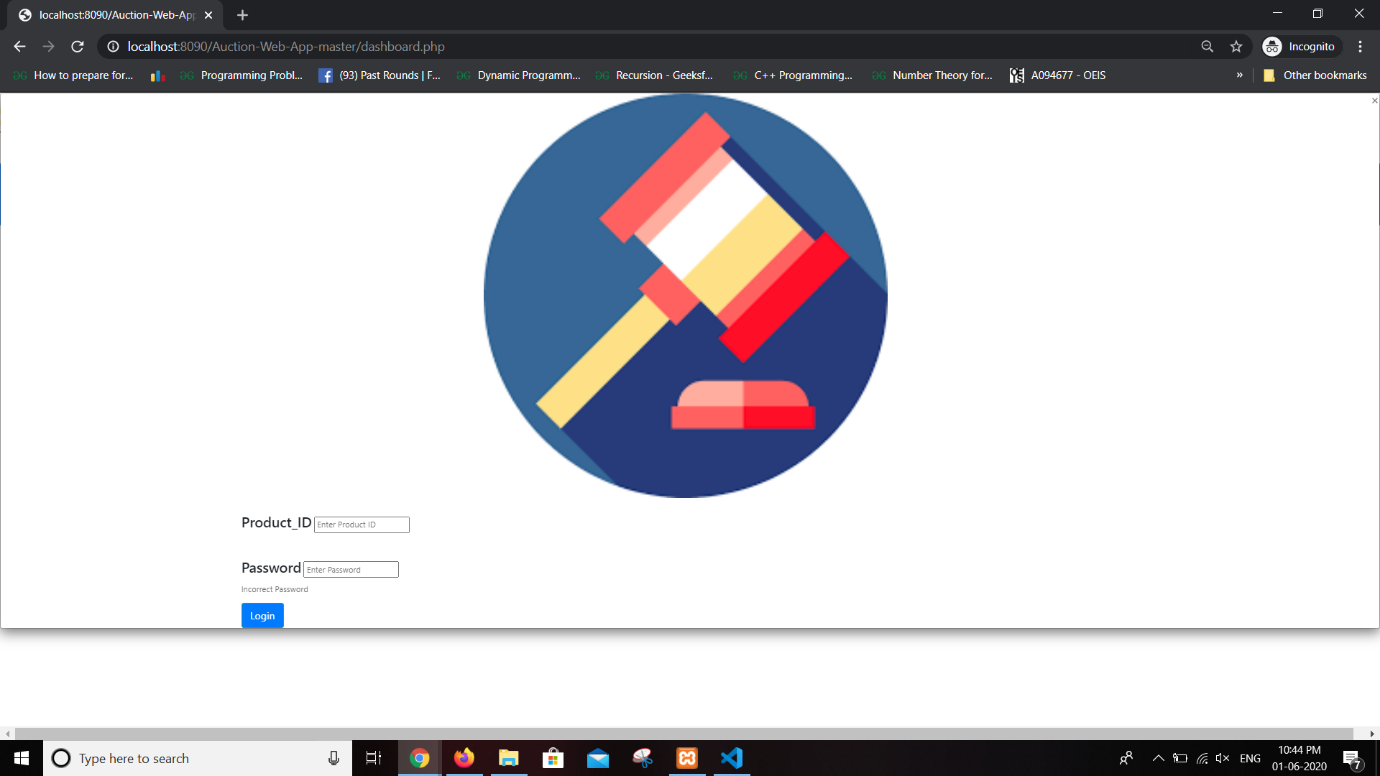
This page will show you the current products that are available for auction. It will show you the name, photo, description, price of the product and you can register for the auction by clicking the button, it will send you the credentials for participating in bid.



Bid login:

File: dashboard.php (modal)

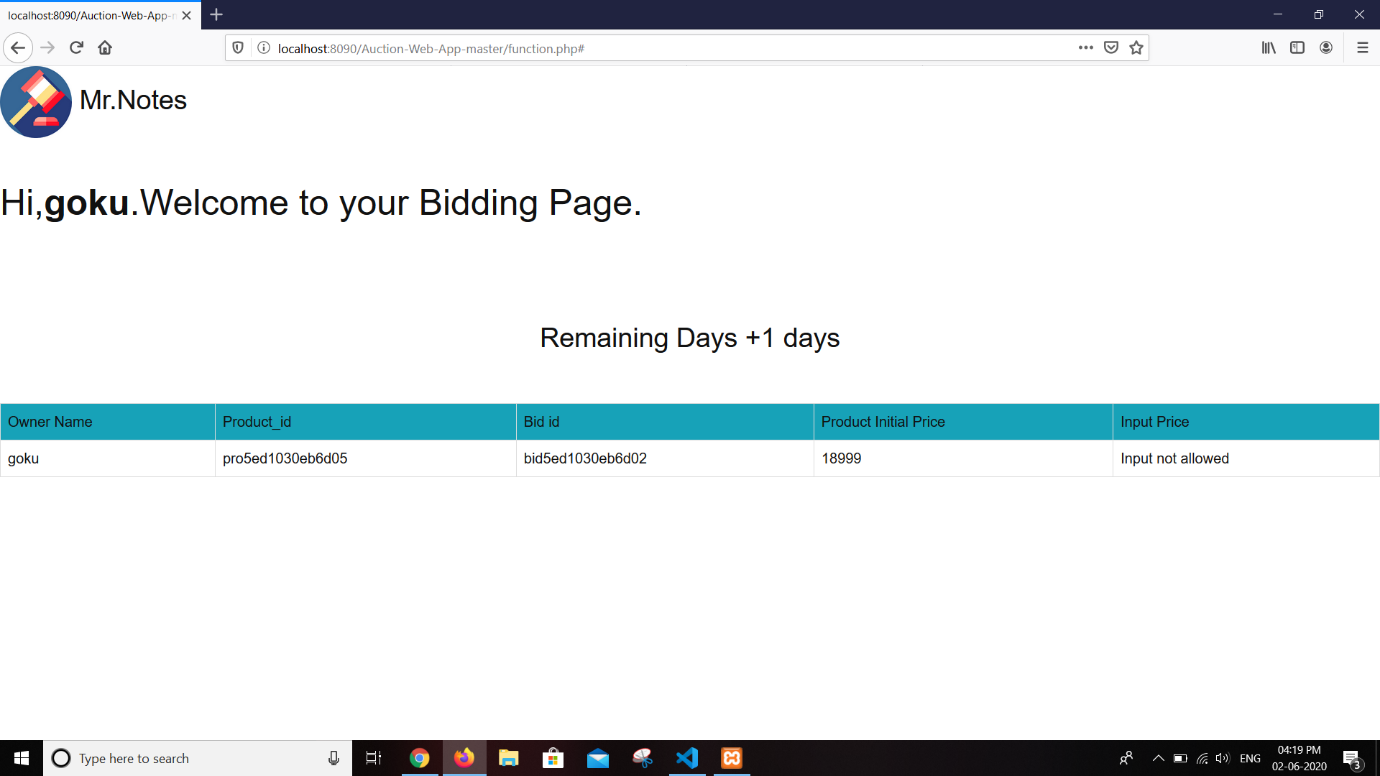
You can enter in bidding process by login through this modal, you have to use credentials provided to you during participate in bid process.

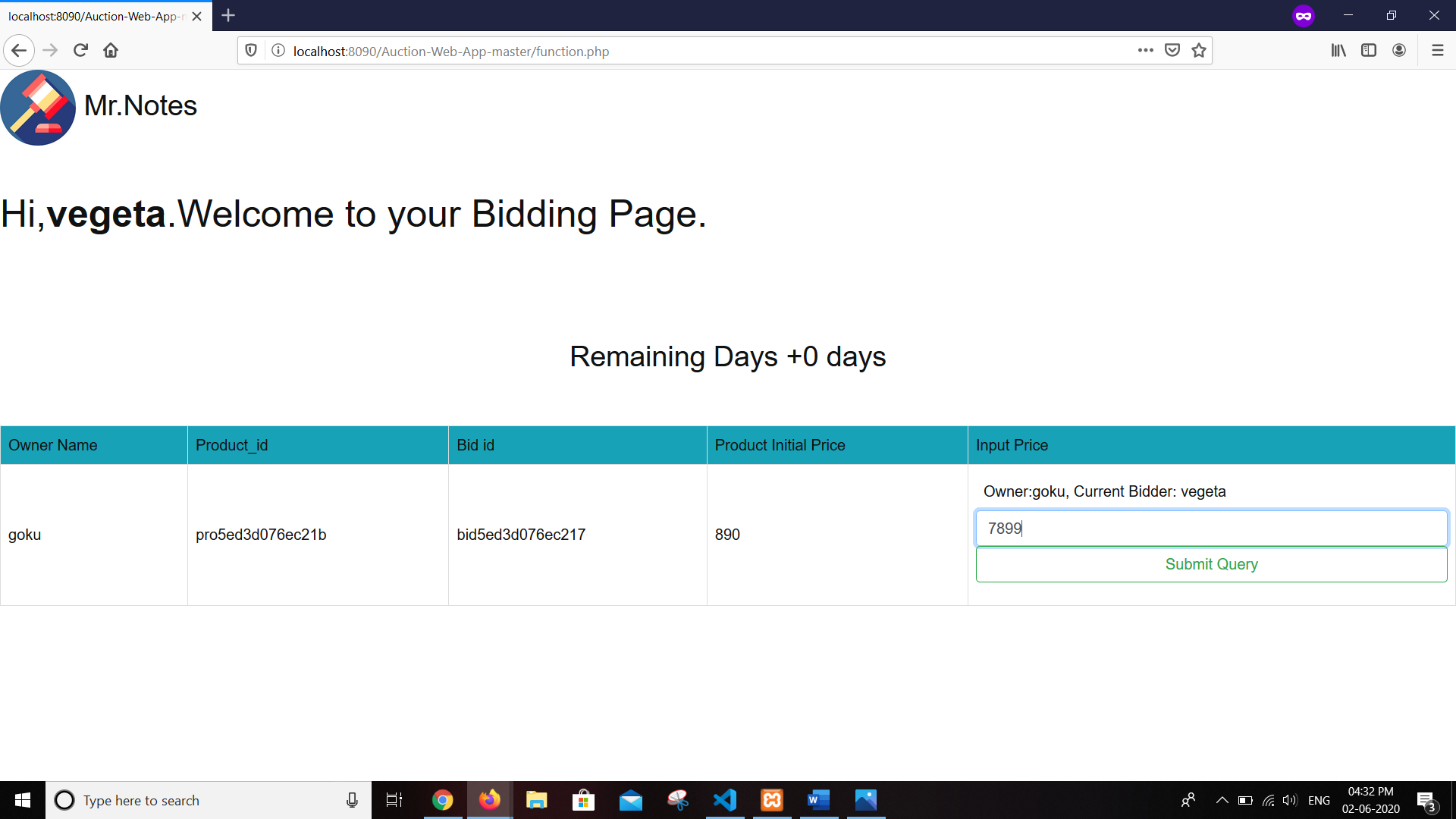


Bidding process:

File: function.php

This is the final page of the website, where user can perform the bidding process, All the details regarding auctioning product is provided on this page, You can enter the amount that you want to bid here. There is constraint that bid amount is always higher than current highest amount or it will show you the error in update section.

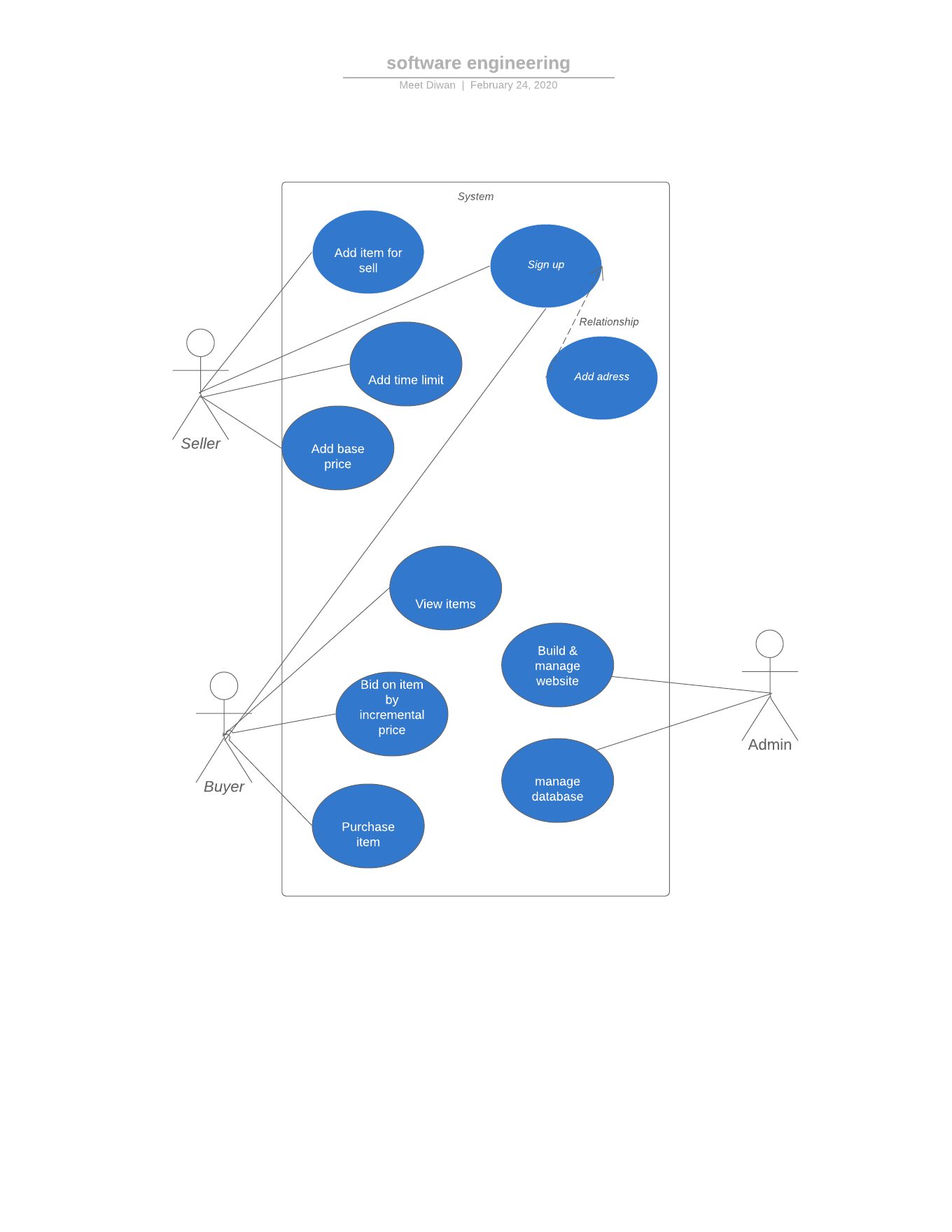




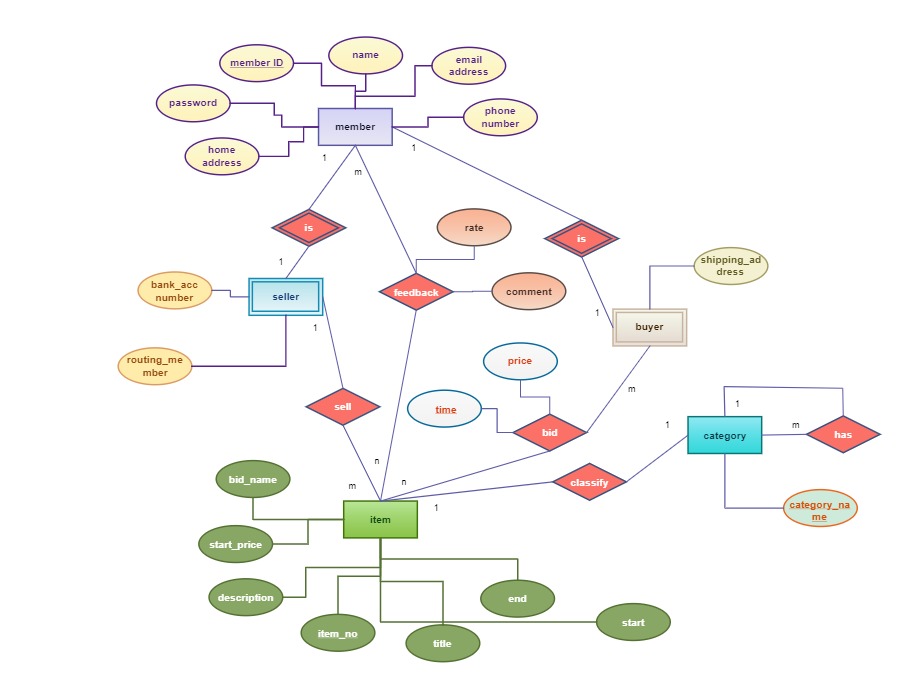
Practical: 5

**Design & Implementation**

**Usecase Diagram:**

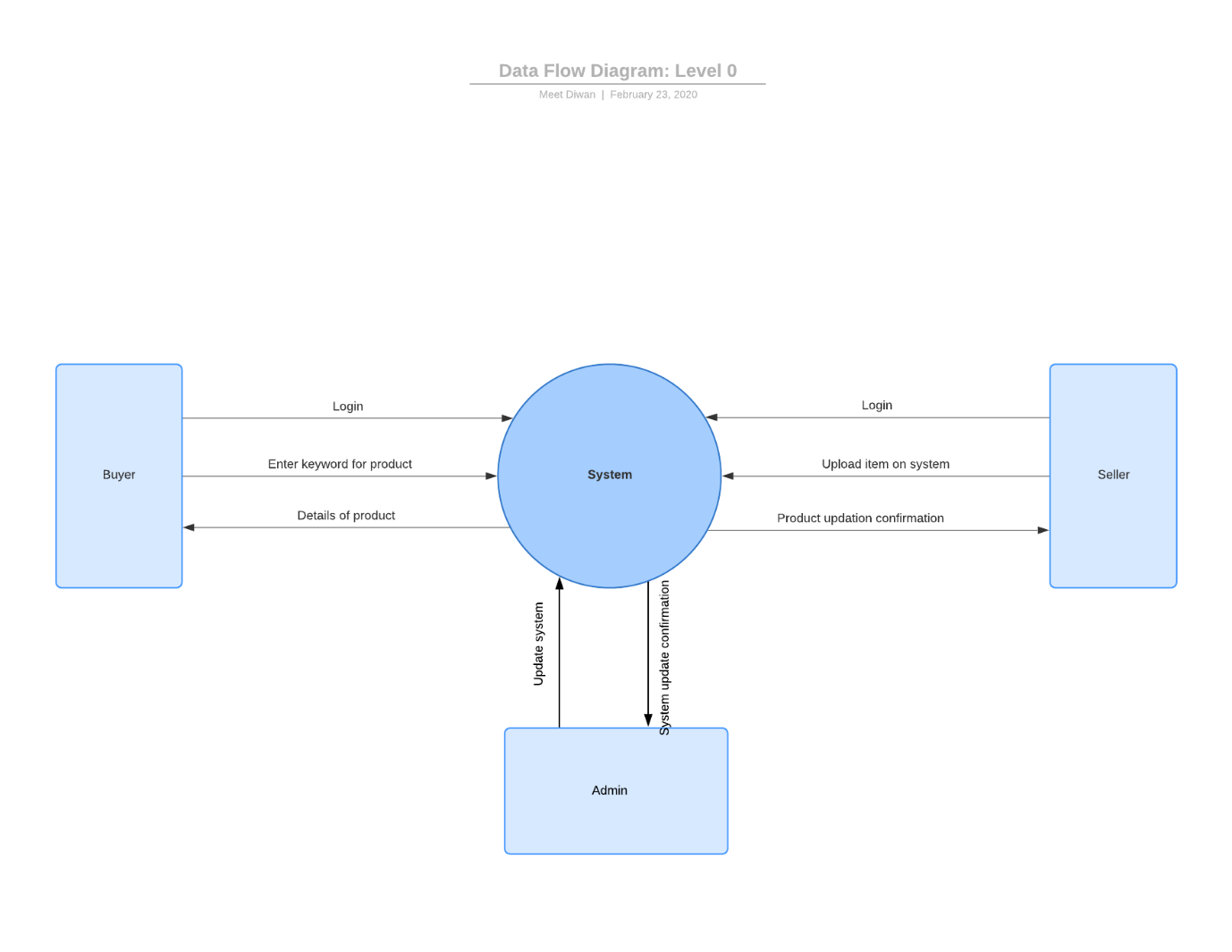


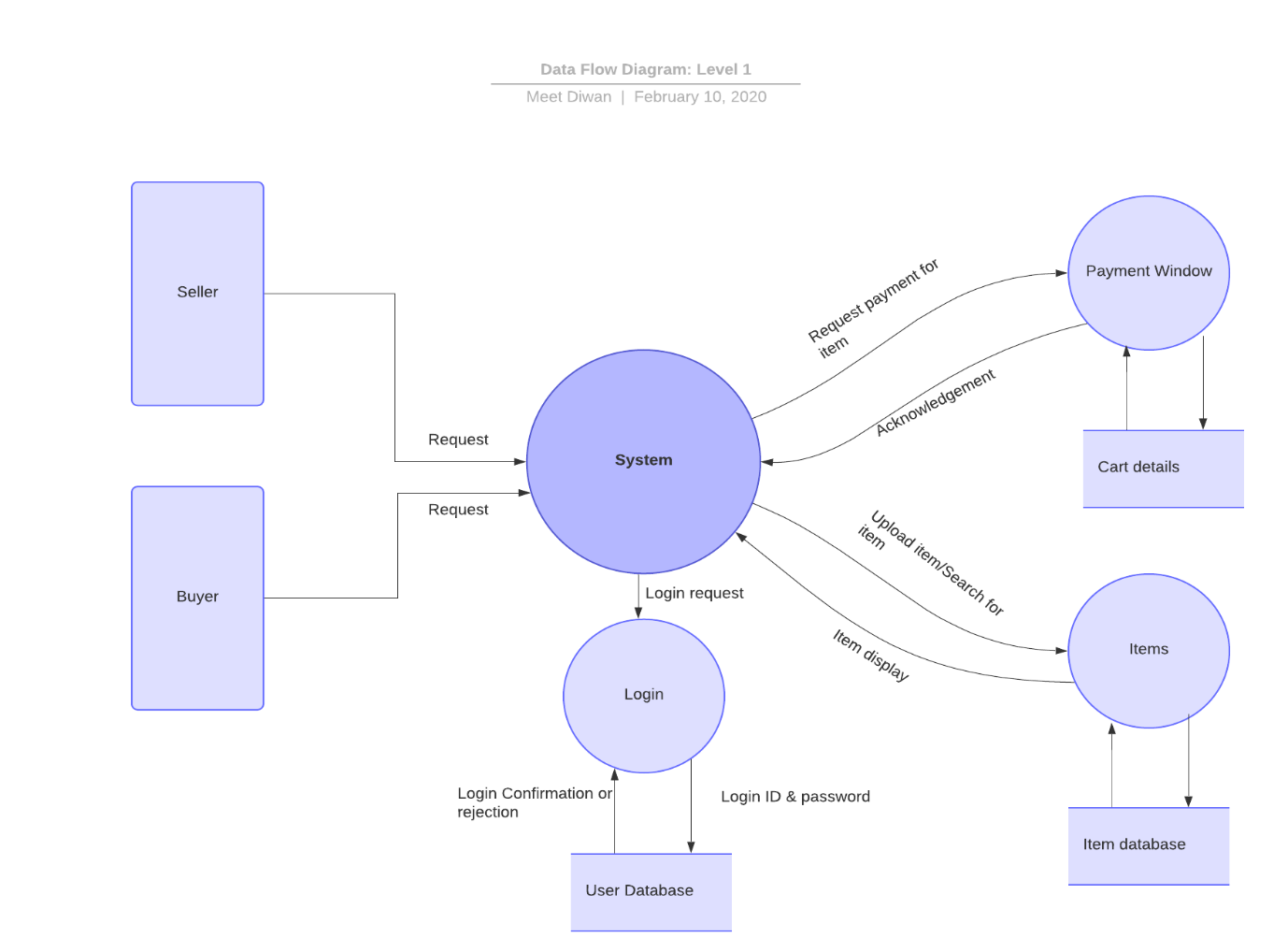
**ER diagram**



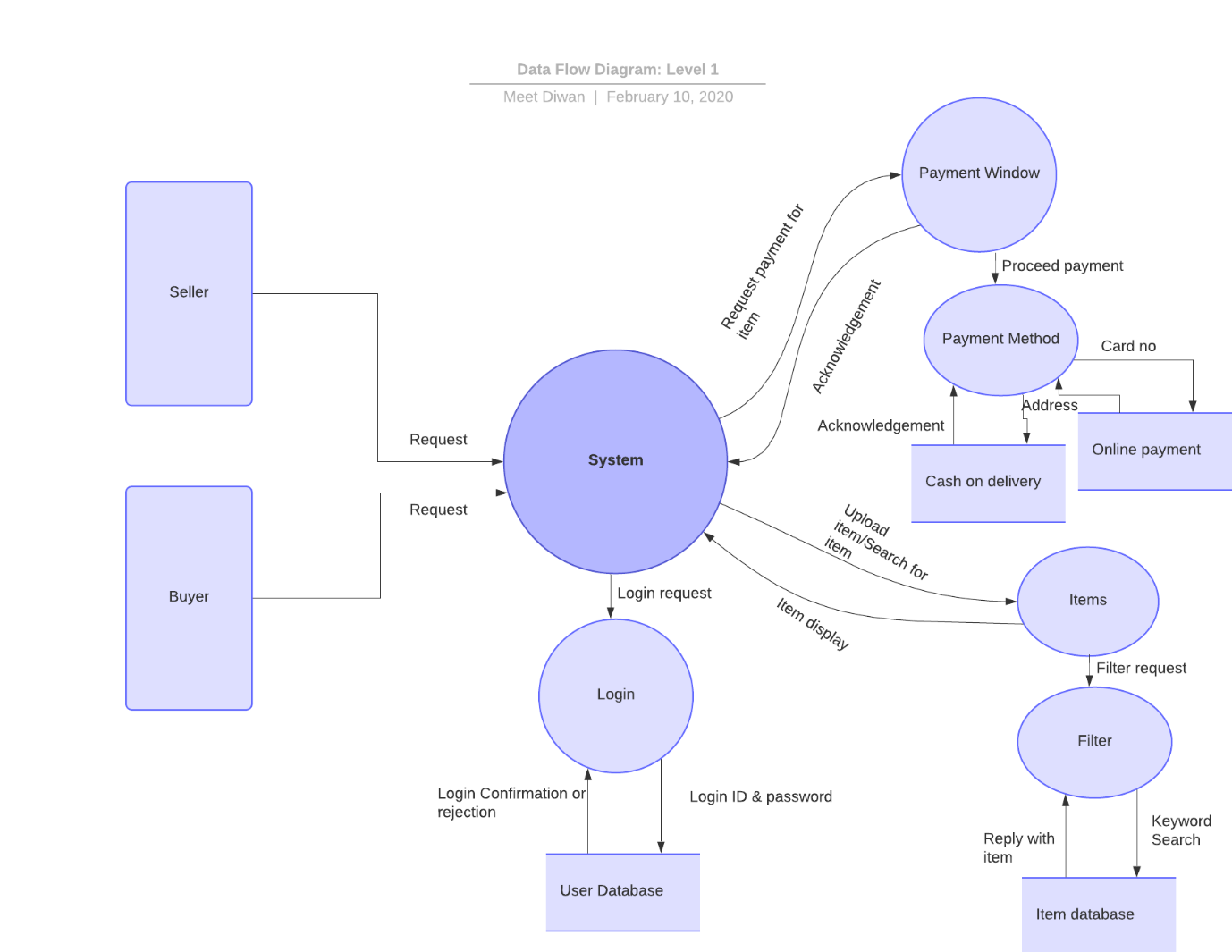
**Data Flow Diagram:**

Level 0:

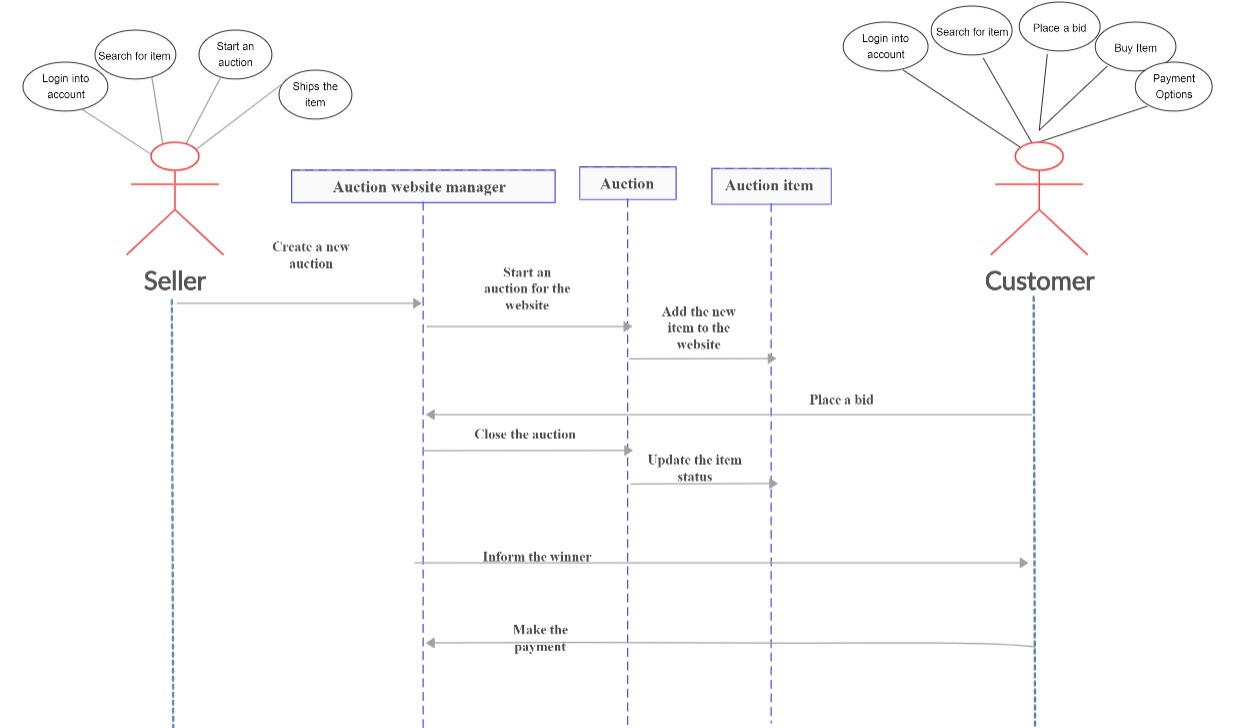
Level 1:



Level 2:



**Sequential Diagram**



**Data Dictionary**

Table Name: Seller or Buyer

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Column Description | Data type | Null value |
| Name | Name of the Seller/Buyer | Varchar | False |
| Id | Unique ID for all seller on the platform | Varchar | False |
| Password | Secure code used by the user for login | Varchar | False |
| Address | Address of the seller/Buyer | Varchar | True |

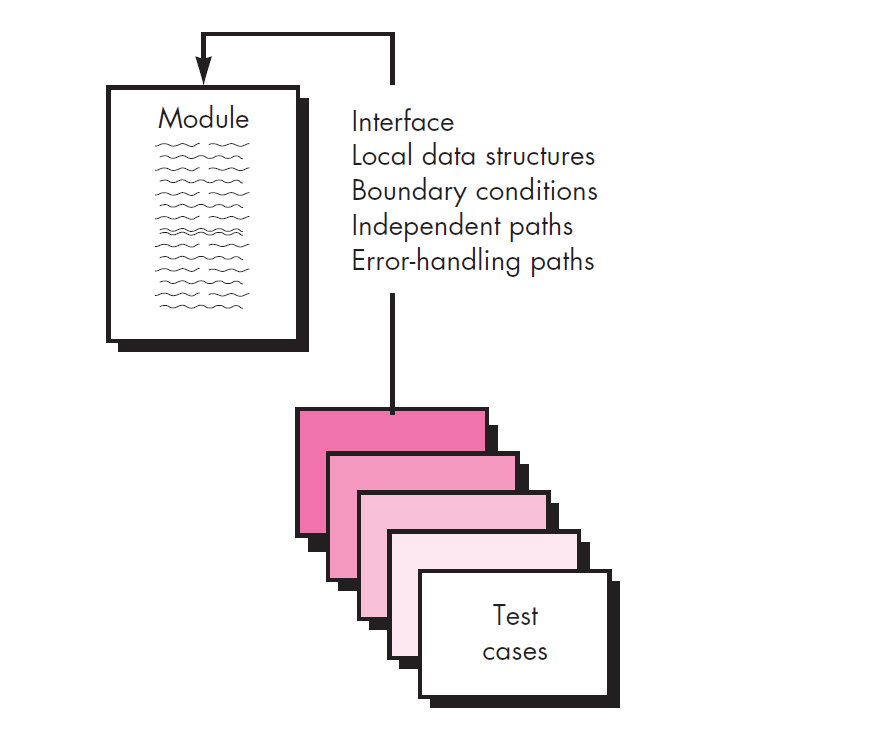
Table Name: Items

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Column Description | Data type | Null value |
| Name | Name of the Item | Varchar | False |
| Id | Unique ID provided to item. | Varchar | False |
| Item Description | Description of item, provided for interested buyers | Varchar | True |
| Price | Price of the item | Integer | True |
| Bid Time | Provided in hours | Integer | True |

Practical : 6

**Unit testing.**

Unit testing focuses verification effort on the smallest unit of software design—the software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and the errors those tests uncover is limited by the constrained scope established for unit testing. The unit test focuses on the internal processing logic and data structures within the boundaries of a component. This type of testing can be conducted in parallel for multiple components.

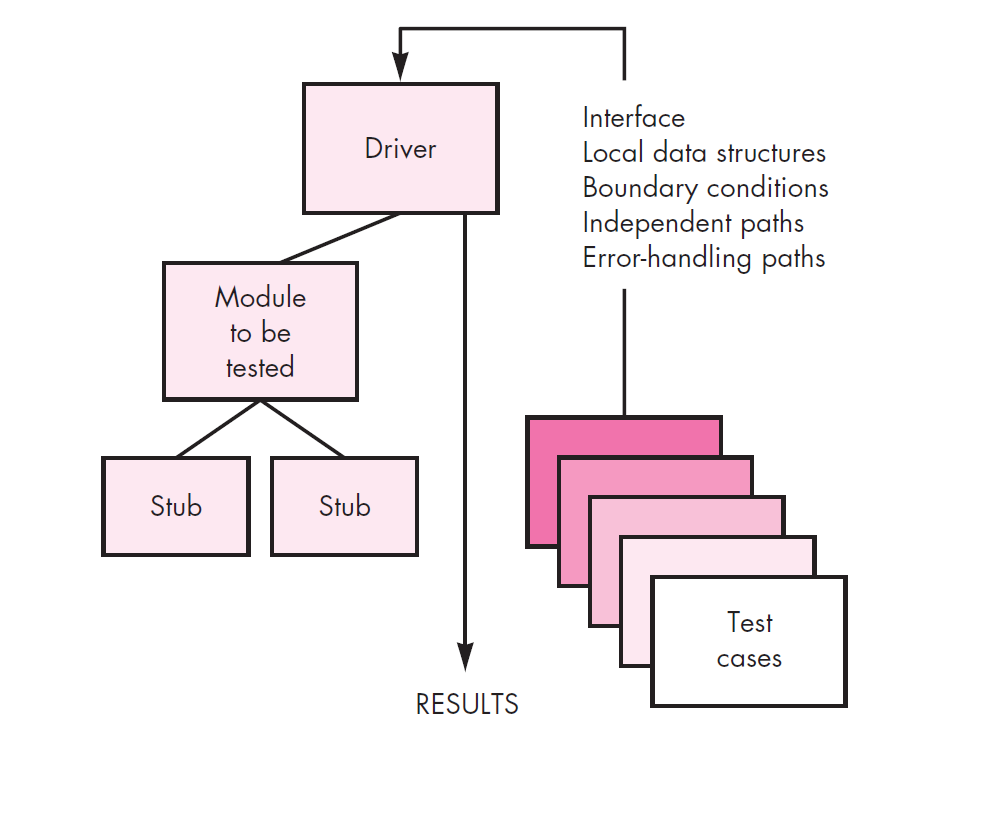


The module interface is tested to ensure that information properly flows into and out of the program unit under test. Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s

execution. All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once. Boundary conditions are tested to ensure that the module operates properly at boundaries established to

limit or restrict processing. And finally, all error-handling paths are tested. Data flow across a component interface is tested before any other testing is initiated. If data do not enter and exit properly, all other tests are moot. In addition, local data structures should be exercised and the local impact on global data should be ascertained

(if possible) during unit testing.



**Cyclomatic Complexity**

Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program. The nodes in the graph indicate the smallest group of commands of a program, and a directed edge in it connects the two nodes i.e. if second command might immediately follow the first command.

M = E – N + 2P

where,

E = the number of edges in the control flow graph

N = the number of nodes in the control flow graph

P = number of nodes that have exit points

**Use of Cyclomatic Complexity :-**

* Determining the independent path executions thus proven to be very helpful for Developers and Testers.
* It can make sure that every path has been tested at least once.
* Thus, help to focus more on uncovered paths.
* Code coverage can be improved.
* Risk associated with program can be evaluated.
* These metrics being used earlier in the program helps in reducing the risks.

**Code –**

IF A = 10 THEN

IF B > C THEN

A = B

ELSE

A = C

ENDIF

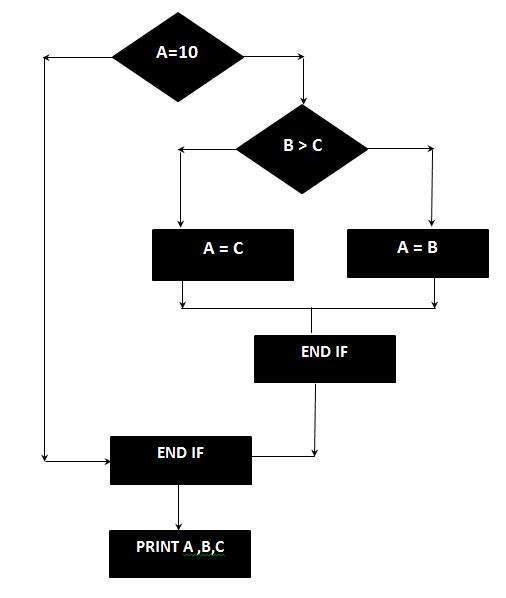
ENDIF

Print A

Print B

Print C

**Control flow graph –**



Total no. of edges: E = 8

Total no. of Nodes: N = 7

M = 8 – 7 + 2p

M = 1 + 2(1)

Hence the cyclomatic complexity is 3 .

Practical 7

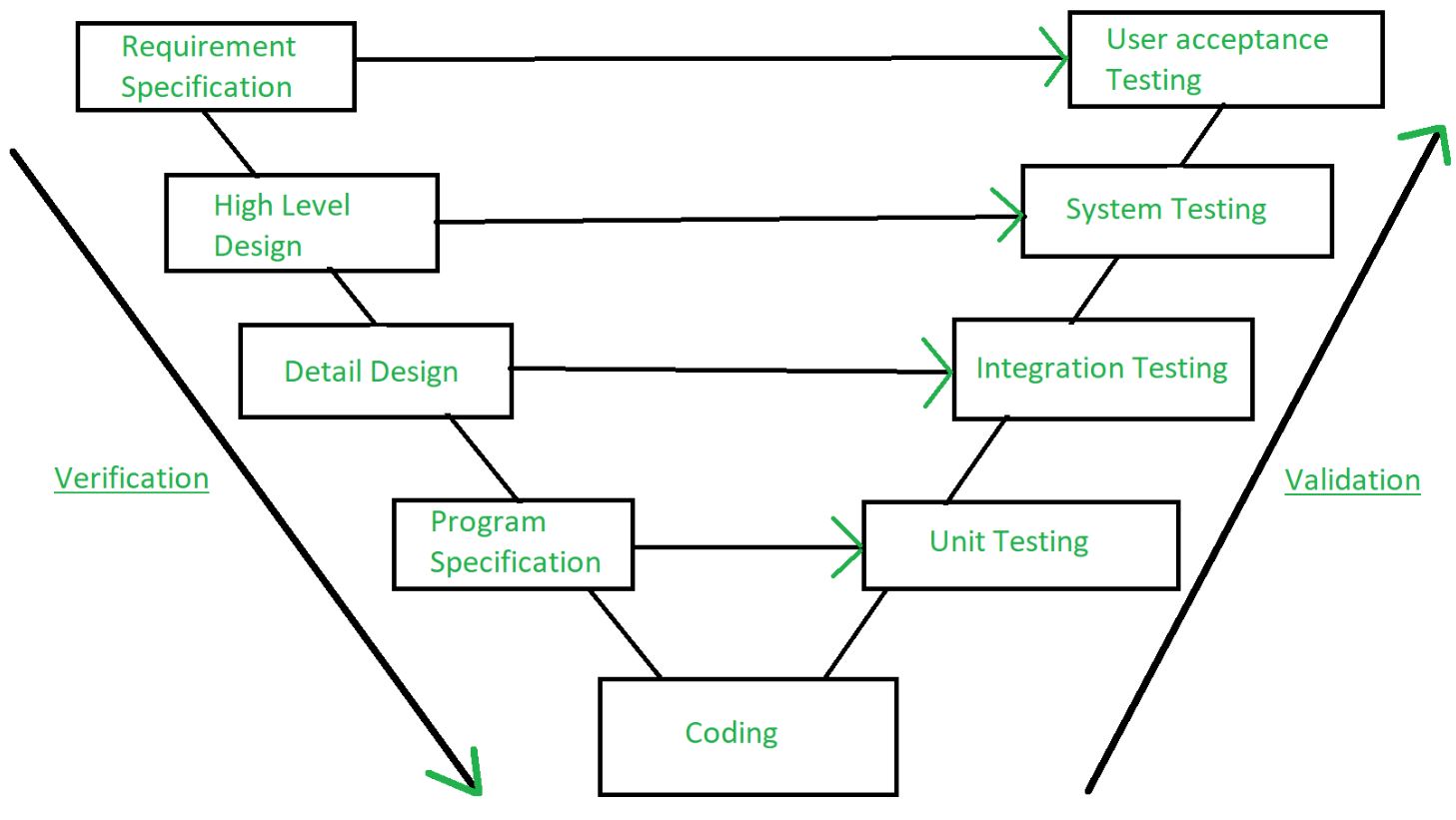
**Validation and Verification**

**Verification** is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfil the requirements that we have. Verification is static testing.

Verification means Are we building the product, right?

**Validation** is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.

Validation means Are we building the right product?



|  |  |
| --- | --- |
| Activities involved in validation:   1. Black box testing 2. White box testing 3. Unit testing 4. Integration testing | Activities involved in verification:   1. Inspections 2. Reviews 3. Walkthroughs 4. Desk-checking |

|  |  |
| --- | --- |
| VERIFICATION | VALIDATION |
| It includes checking documents, design, codes and programs. | It includes testing and validating the actual product. |
| Verification is the static testing. | Validation is the dynamic testing. |
| It does not include the execution of the code. | It includes the execution of the code. |
| Methods used in verification are reviews, walkthroughs, inspections and desk-checking. | Methods used in validation are Black Box Testing, White Box Testing and non-functional testing. |
| It checks whether the software conforms to specifications or not. | It checks whether the software meets the requirements and expectations of a customer or not. |
| It can find the bugs in the early stage of the development. | It can only find the bugs that could not be found by the verification process. |
| The goal of verification is application and software architecture and specification. | The goal of validation is an actual product. |
| Quality assurance team does verification. | Validation is executed on software code with the help of testing team. |
| It comes before validation. | It comes after verification. |

**Alpha and Beta Testing**

* **Alpha testing**

– Conducted at the developer’s site by end users

– Software is used in a natural setting with developers watching intently

– Testing is conducted in a controlled environment

* **Beta testing**

– Conducted at end-user sites

– Developer is generally not present

– It serves as a live application of the software in an environment that cannot be controlled by the developer

– The end-user records all problems that are encountered and reports these to the developers at regular intervals

* After beta testing is complete, software engineers make software modifications and prepare for release of the software product to the entire customer base.

Practical: 8

SRS Document: Paytm

1. **Introduction**
   1. **Purpose:**

To provide digital platform for money transfer, paying bill & other activities related to money transfer like Bank to bank transfer, UPI facility etc.

* 1. **Document conventions:**

I have used Times New Roman fonts in this document with font size of 18 and heading with Bold letters.

* 1. **Intended audience:**

Intended audiences are programmer, researcher or any software engineer who wants details regarding platform like Paytm.

* 1. **Project scope:**

This project model is operating worldwide. Users from any age, from any background needs this kind of platform for money transfer.

* 1. **Reference:**

*Pressman, A practitioner’s approach* for prototype of SRS document.

1. **Overall description**
   1. **Product perspective:**

It is available on both platforms. Website and Application(iOS/Android).

* 1. **Product feature:**
* Money transfer
* Local wallet
* Bill payment
* Train ticket booking
* Movie ticket booking
* UPI facility
* Bank account connection
* Passbook
* Pay using QR code
* Mobile recharge
  1. **Operating environment**

Both mobile application and Website facility available.

* 1. **Assumptions & dependencies**

User should have connected to Internet.

User must have electronic device like mobile or laptop.

User should have bank account

User must verify KYC requirement according to government guidelines

1. **System features**
   1. **Feature: money transfer**

Money can be transfer in 3 modes

1. Bank to wallet
2. Wallet to wallet
3. Wallet to bank

User must have connected to bank account to start using payment feature

* 1. **Feature: Bill payment**

This is the basic needs for every user of paying bill for electricity, gas, water etc.

* 1. **Feature: Reservation**

User can reserve seat for trains and movie theatre using this platform

1. **External Interface requirement**
   1. **User Interface**

Works on all platforms.

Examples are iOS, Android, Windows

* 1. **Hardware Interface**

Mobile, Personal computer, laptop having internet facilities

1. **Non-functional Requirements**
   1. **Performance requirements**

3G/4G internet network recommended

* 1. **Security requirement**

User must not share their login credentials to anyone. One Time Password(OTP) is only for approving payments, do not share it with anyone.

**Practical-9**

**ISO 27001:2005 ISMS security Policy with respect to any Software organization.**

**PCIDSS, Sarbanes oxley act 2002.**

**Introduction**

**ISO/IEC 27001:2013** is an information security standard that was published in September2013 It supersedes ISO/IEC 27001:2005, and is published by the International Organization for Standardization (ISO) and the International Electro technical Commission (IEC) under the joint ISO and IEC subcommittee, ISO/IEC JTC 1/SC 27. It is a specification for an Information Security Management System (ISMS). Organizations which meet the standard may be certified compliant by an independent and accredited certification body on successful completion of a formal compliance audit.

**Structure of the standard**

The official title of the standard is "Information technology - Security techniques - Information Security Management Systems - Requirements".

ISO/IEC 27001:2013 has ten short clauses, plus a long annex, which cover - -

1. Scope of the standard
2. How the document is referenced
3. Reuse of the terms and definitions in ISO/IEC 27000
4. Organizational context and stakeholders
5. Information security leadership and high-level support for policy
6. Planning an [information security management system;](https://en.wikipedia.org/w/index.php?title=Information_security_management_system&action=edit&redlink=1) risk assessment; risk treatment
7. Supporting an information security management system
8. Making an information security management system operational
9. Reviewing the system's performance
10. Corrective action

**Network Security**

Network security assumes importance to the organization when viewed in light that networks change frequently as new users and devices are added and newer data communication technologies are introduced, usage of various networking, communications, and computing technologies to effectively meet the expanding need, sensitive data is increasingly transmitted over networks, proliferation of internet access has increased the vulnerability as employees use internet more for information and knowledge.

The primary objectives of a network security policy should be to ensure that access to company’s network is only provided to authorized users, that adequate controls are in place to manage remote users, that all equipment can be recognized uniquely, that networks should be segregated based on needs, and that appropriate network routing protocols are enabled.

Typical policy statements for Network Security include :-

1. Appropriate authentication mechanisms shall be used to control the access by remote users.
2. Allocation of network access rights shall be provided as per the business and security requirements
3. Two-factor authentication shall be used for authenticating users using mobile/remote systems

**Purpose of Information Security Policy**

In many cases, the executives have no idea as to how information security can help their organization, so the main purpose of the [policy](https://advisera.com/27001academy/documentation/information-security-policy/?icn=paid-document-27001-information-security-policy&ici=top-policy-txt) is that the top management defines what it wants to achieve with information security.

The second purpose is to create a document that the executives will find easy to understand, and with which they will be able to control everything that is happening within the ISMS – they don’t need to know the details of, say, risk assessment, but they do need to know who is responsible for the ISMS, and what to expect from it.

**What should the Information Security Policy contain?**

ISO 27001 doesn’t say too much about the policy, but it does say the following :-

1. The policy needs to be adapted to the organization – this means you cannot simply copy the policy from a large manufacturing company and use it in a small IT company.
2. It needs to define the framework for setting information security objectives – basically, the policy needs to define how the objectives are proposed, how they are approved, and how they are reviewed. See also: ISO 27001 control objectives – Why are they important?
3. The policy must show the commitment of top management to fulfil the requirements of all interested parties and to continually improve the ISMS – this is normally done through a kind of a statement within the policy.
4. The policy must be communicated within the company, but also – where appropriate – to interested parties; best practice is to define who is responsible for such communication, and then that person is responsible for doing it continuously.
5. The policy must be regularly reviewed – an owner of a policy should be defined, and this person is responsible for keeping the policy up to date.

**Inputs needed :**

There are a couple of inputs you have to take into account when writing the policy:

1. Top management intentions with information security – the best thing would be to schedule an interview with your CEO and go through all the elements of the policy; you might send him an email a couple of days before the meeting, so that he has time to think about it.
2. Legislation and contractual requirements – your policy should reflect those.
3. Existing system for setting objectives – if such system exists, you should refer to it.

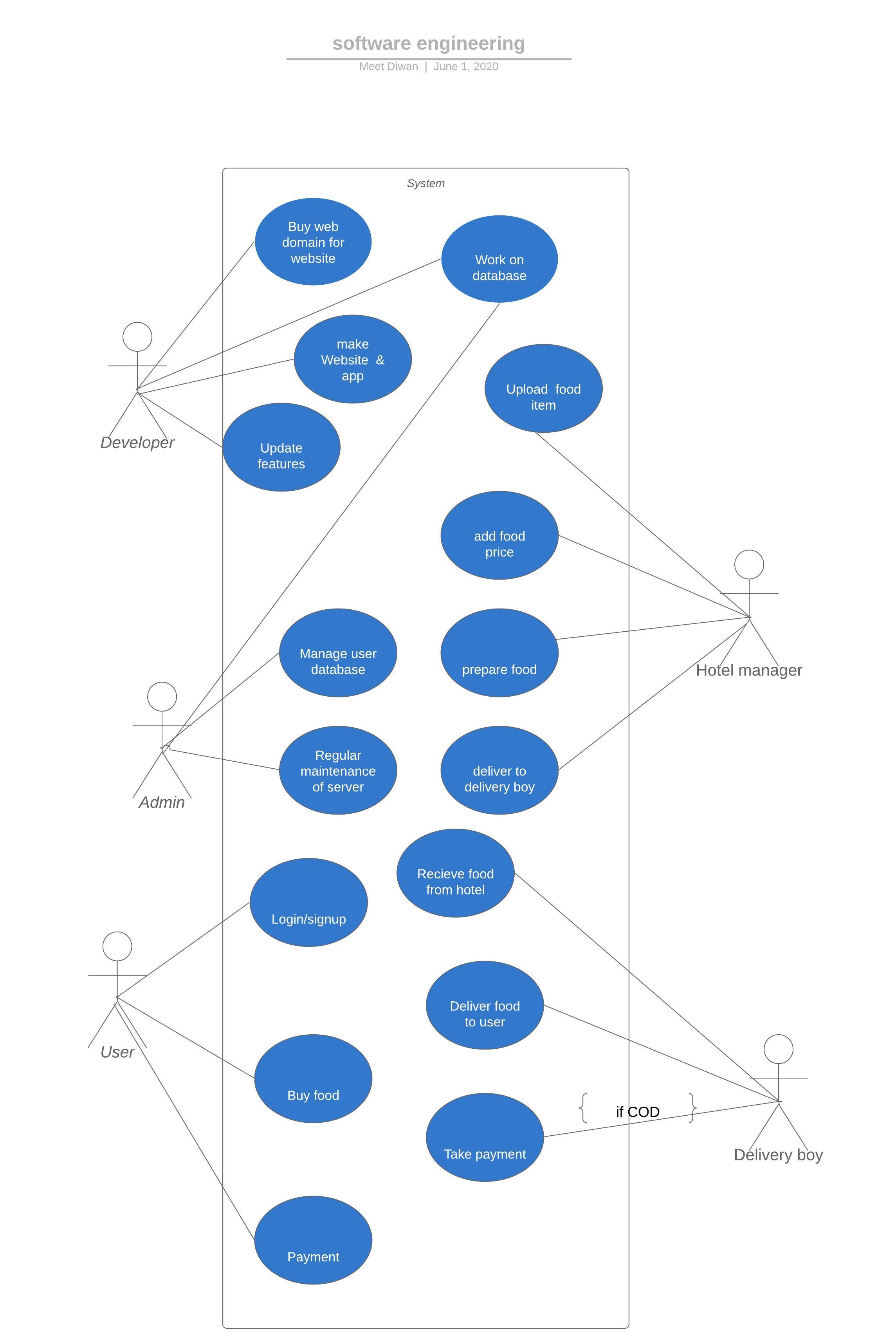
**Practical-10**

**Compare Zomato, Swiggy, Delfoo application with their unique feature and demonstrate their UML diagrams.**

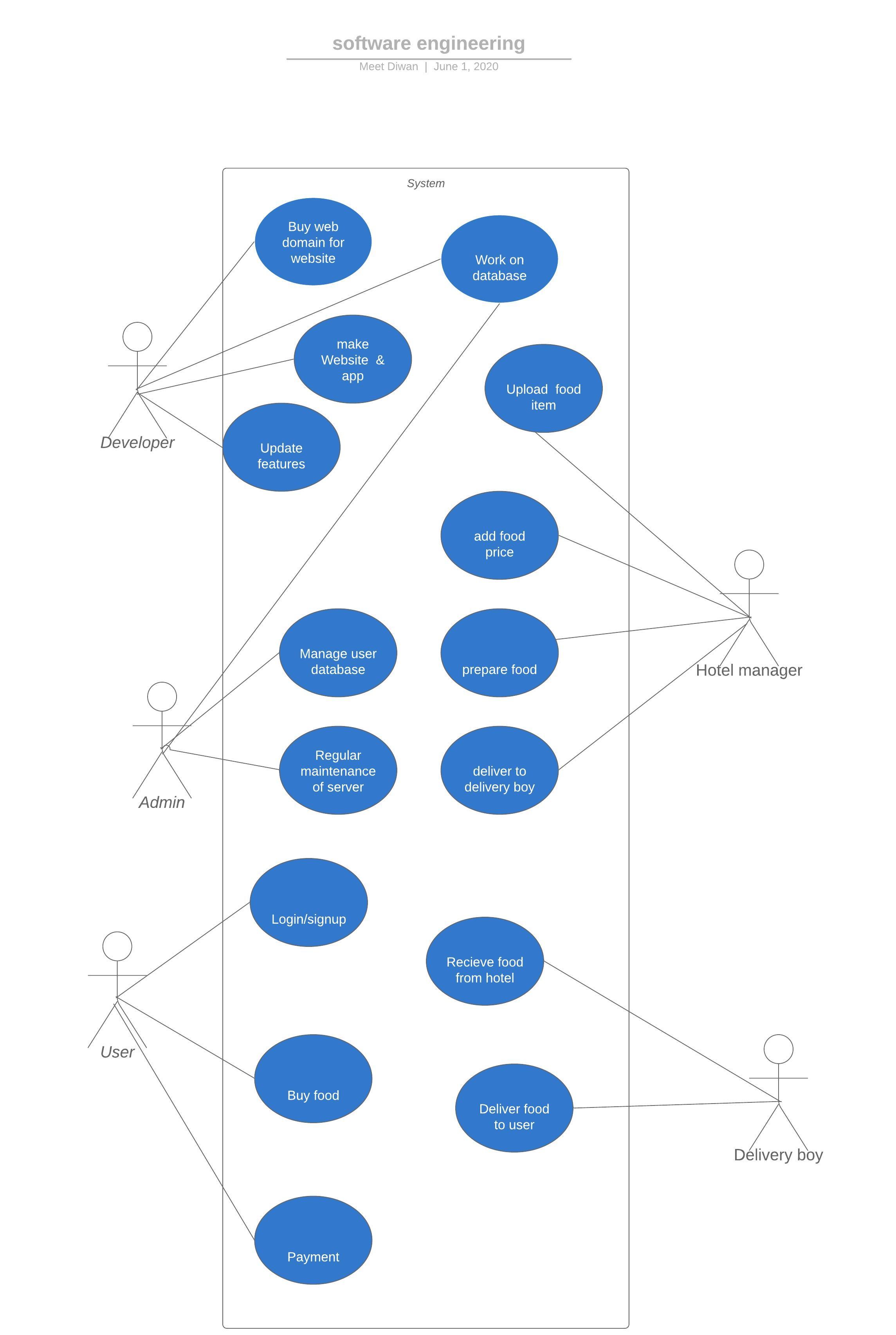
Zomato, Swiggy and Delfoo are online platforms that provides food home delivery. Zomato and Swiggy are same kind of platform having features like order food, rate delivery, track order etc. One unique feature of Delfoo app is it provides multiple restaurants food order and deliver it together. Like example If you want to order from Domino’s and La’pinoz which are on different side of city, Delfoo gives you feature to order from Domino’s and La’pinoz both together.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Feature | Zomato | Swiggy | Delfoo |
| 1 | Food order | Yes | Yes | Yes |
| 2 | Online Payment | Yes | Yes | No |
| 3 | Menu updation | Yes | Yes | Yes |
| 4 | Cash on delivery | Yes | No | Yes |
| 5 | Order from multiple restaurant on single order | No | No | Yes |
| 6 | Digital invoicing | Yes | Yes | No |
| 7 | Multiple platform support (Android / iOS) | Yes | Yes | Yes |
| 8 | Meal subscription (Tiffin service) | No | No | Yes |

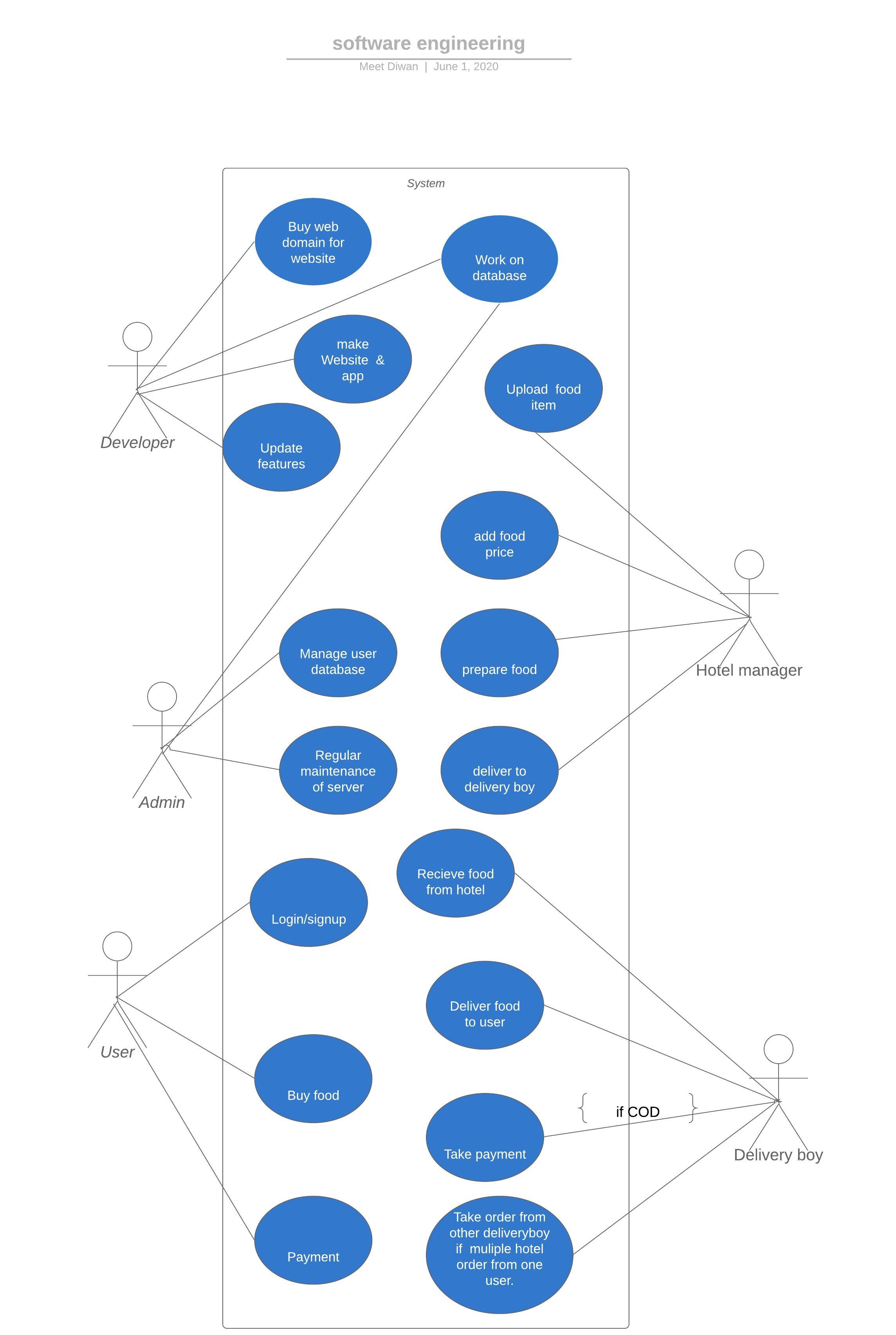
**Zomato Use case:**

****

**Swiggy Use-case diagram:**

****

**Delfoo use-case diagram:**

****

1. **OEP**

**Prepare SRS document for application or web**

**interface which give you distinct feature like RAIL**

**YATRI.**

1. **Introduction**

1.1 **Purpose**:

To provide functionalities of booking Indian railway tickets, checking availability of seats in train, check current status of the train etc.

1.2 **Document conventions:**

I have used Times New Roman fonts in this document with font size of 16 and heading with Bold letters.

1.3 **Intended audience**:

Intended audiences are programmer, researcher or any software engineer who wants details regarding platform like RailYatri.

1.4 **Project scope**:

This platform is in use all over India, users above age 18 can use more functionality of this platform for booking tickets online, users above age14 can use this application to track the status of train.

1.5 **Reference**:

Pressman, A practitioner’s approach for prototype of SRS document.

**2. Overall description**

**2.1 Product perspective:**

It is available on both platforms. Website and Application(iOS/Android).

**2.2 Product feature:**

* Book train ticket
  + One can book ticket using IRCTC account
* Book bus ticket
  + New InterCity project of RailYatri make online bus ticket booking available
* PNR Status
  + You can check details of your journey train using PNR status
* Railway timetable
  + Arrival & Departure time, and platform details are available on app.
* Live status of trains
  + Current location of the train, how far train from particular station functionality is available on RailYatri.
* Seat availability in trains
  + This is the prominent feature in any app like RailYatri, passengers always like to check seat availability before making payment online.
* Wallet facility
  + RailYatri has its own wallet, which can be used to make online payment, RailYatri gives cashbacks in this wallet than can be used in next journey.
* Food on train
  + One can order food from train using PNR number. One can select upcoming station as pickup point.

**2.3 Operating environment**

Both mobile application and Website facility available.

* 1. **Assumptions & dependencies**

User should have connected to Internet.

User must have electronic device like mobile or laptop.

User must have IRCTC account to book a ticket of train.

**3. System features**

**3.1 Feature: Seat reservation**

User can book both train and bus ticket via RailYatri app. User must have registered to IRCTC, in accordance to book Indian Railway ticket.

**3.2 Feature: Train status**

User can track any train using this feature. This feature lets user get tiny information like how many time ago train left previous station, estimated time to reach next station.

**4. External Interface requirement**

**4.1 User Interface**

Works on all platforms.

Examples are iOS, Android, Windows

**4.2 Hardware Interface**

Mobile, Personal computer, laptop having internet facilities

**5. Non-functional Requirements**

**5.1 Performance requirements**

3G/4G internet network recommended for using this app.

24\*7 support from service staff because trains are running for 24 hours.

Accurate info of current train status.

**5.2 Security requirement**

If any user provides their account credentials to anyone then anyone can use their wallet for booking tickets.