**High level Architecture**

**<05>:<INSTASHOP>**

**<team member names & ids>**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **23100268** | **FARVA TALIB** |
| **23100233** | **MUHAMMAD UMAIR MOHSIN** |
| **23100091** | **MUHAMMAD BILAL SHAHID** |
| **23100254** | **SALMAN MASOOD** |
| **23100016** | **MUHAMMAD AFFAN ASHRAF** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Architecture diagram | 30 | 24 |
| Architecture description | 10 | 8 |
| Architecture justification | 15 | 8 |
| Risk Management | 20 | 20 |
| Tools & Technologies | 10 | 10 |
| Hardware Requirements | 5 | 5 |
| Who did what | 5 | 5 |
| Review checklist | 2 | 2 |
| Overall formatting/template | 3 | 1 |
| Late submission penalty | -20 |  |
| **Total** | **100** | **83** |
| Review | 20 |  |
| **Grand Total** |  |  |

**Table of Contents**

[1.](#_heading=h.gjdgxs) Introduction 3

[2.](#_heading=h.30j0zll) System Architecture 4

[2.1 Architecture Diagram 4](#_heading=h.1fob9te)

[2.2 Architecture Description 5](#_heading=h.3znysh7)

[2.3 Justification of the Architecture 6](#_heading=h.2et92p0)

[3.](#_heading=h.tyjcwt) Risk Management 7

[3.1 Potential Risks and Mitigation Strategies 7](#_heading=h.3dy6vkm)

[4.](#_heading=h.1t3h5sf) Tools and Technologies 8

[5.](#_heading=h.4d34og8) Hardware Requirements 9

[6.](#_heading=h.2s8eyo1) Who Did What? 10

[7.](#_heading=h.17dp8vu) Review checklist 10

# Introduction

<Give an overview of the project here. The overview must highlight the overall objectives of the project and its potential users and customers.>

InstaShop will be a web-based portal which aims to revolutionize influencer marketing. For those who may be unaware, influencer marketing is a recent development in the digital world which involves a brand collaborating with an online influencer to market one of its products or services. These influencers are usually found on social media platforms (Instagram and Snapchat) and have a decent number of followers to whom they market a particular brand’s products or services.

There are two main parties involved in this process, the client and the influencer. For better understanding of the motivation behind this project, we must first walk through the process that is generally followed in influencer marketing from the perspective of both the client and the influencer.

For the client, the first step is perhaps the most tedious which is related to finding the right influencer for your brand. This is usually achieved by manually searching social media platforms or relying on word of mouth from friends/family regarding a particular influencer. The second step is contacting the shortlisted profiles. This is either done through direct messages or emails, both of which are again tedious tasks with no guarantees of a timely response. The final step (assuming the contract has been fulfilled) is the issue of payment. This is perhaps the most troublesome aspect because there is no guarantee that the influencer will produce content that is upto the mark and not run off with any advance payments made.

For the influencer (unless they have a huge following) it is usually hard to find clients for collaborations or for sponsored content. Since these influencers are very active on social media, their inbox and comment sections are almost always flooded which means they tend to miss out on potential business opportunities just because they weren’t able to see the direct message. Some profiles do have designated emails for business inquiries but their response times are in most cases not ideal because they just don’t check their email that often. Finally, the issue of payment also exists. There is no guarantee that the client will pay the influencer the full amount in a timely manner even if the work has been done upto the client’s standards and deadline.

As visible, both of these parties are in dire need of a platform that could automate most of these tasks for them as well as provide payment guarantees. This is where InstaShop comes in. It aims to streamline all the steps involved in this process by providing an easy to use web application thus saving time and effort for both parties involved. For the client, it makes it easier to search relevant influencers courtesy of our database and filtering method, connect with shortlisted influencers (via email or live chat) and have your payment secured (via escrow).

For the influencer, you essentially get access to a marketplace where you can find potential clients, not miss out on potential business opportunities just because your inbox was too cluttered and have a guarantee that the client will pay you for your work.

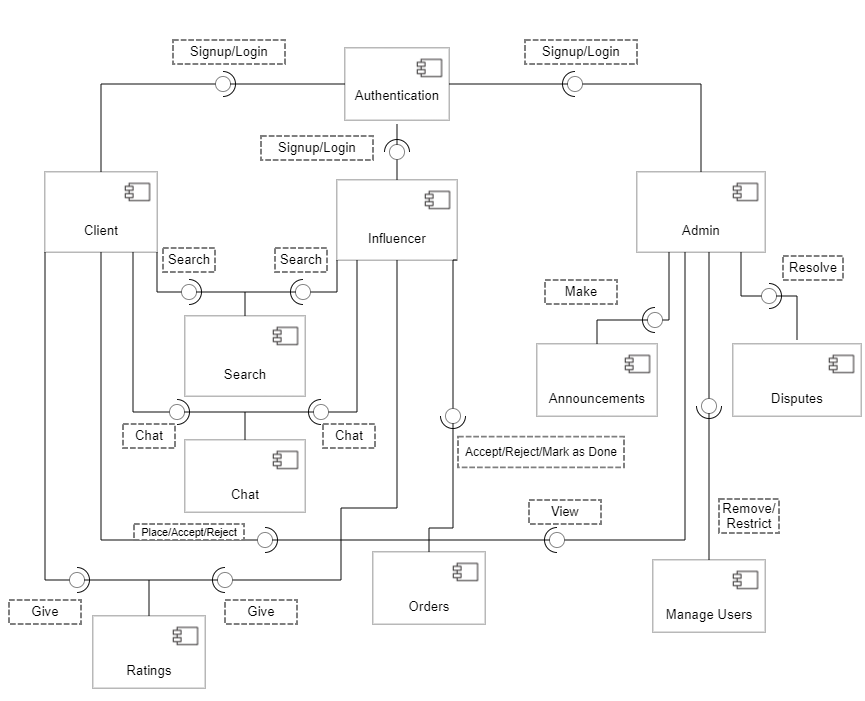
# System Architecture

## Architecture Diagram

<Draw a diagram of the system architecture. Remember that your system’s architecture may be defined using multiple architectural patterns.>

[The components that you have identified should also be reflected in the layers, i.e., which layer/subsystem will contain which components. Without this, your system architecture diagram remains general—it should be specific to your system.]

System Architecture Component Diagram



# 

# 

## Architecture Description

<Give description of each subsystem in the architecture diagram above. Moreover, give description of how subsystems interact with each other>

**Interaction of Subsystems with each other**

The main architectural pattern that is followed is the MVC pattern. The Model View Controller architectural pattern separates concerns into one of 3 buckets:

1- **Model**: Stores and manages data. It is the application’s dynamic data structure (usually a database) which is independent of the user interface. It directly manages the data, logic and rules of the application. In other words, it is responsible for managing the data for the application. It receives user input from the controller.

2- **View**: Graphical User Interface. The view renders the presentation of the model in a particular format. It is a visual representation of the data- like a chart, diagram, table, form. It contains all functionality that directly interacts with the user like clicking a button or an enter event.

3- **Controller**: The controller connects the model and view. The controller converts inputs from the view to demands to retrieve/update data in the model.  
The controller receives input from view, uses logic to translate the input to a demand for the model, the model grabs the data, the controller passes data from the model back to the view for the user to see in a nice display.

**Description of Subsystems**

The architecture of the web application is designed in such a manner that each subsystem is independent of another and only interacts with the actors. This is done on purpose so that these subsystems can be developed in an isolated environment and be made more general by providing APIs and endpoints. For example, the authentication module can be used for any kind of application (web or mobile) by making small edits in only the structure of the inputs received

**Authentication**

The authentication module would be responsible for providing the first layer of security regarding actors who can access and use the system. It is connected to all 3 main actors as all of them are subject to authentication upon Signup and Login.

**Search**

The search module is responsible for providing the features of searching the database

and finding the relevant results. The main users of this subsystem would be the Clients and Influencers.

**Chat**

The chat module is responsible for providing communication between the two actors, client and influencer.

**Order**

The order module is perhaps the most complicated subsystem. It would also comprise of the functionality of the “Contracts” as well as some incorporation of an external payment module.

**Rating**

The rating module is responsible for providing the “Rating and reviews” functionality that the platform offers. It interacts with two actors, the client and the influencer. This functionality is crucial to maintain the integrity and quality of people using the platform

**Dispute**

The dispute module is strictly a subsystem to be used by the admin. It will provide the functionality of solving disputes whenever they arise.

**Announcements**

The announcements module is relatively simple for now as it only provides the feature of sending an announcement by the admin to the other two actors. However, in future this subsystem can be used for other advance features as well hence the separate module entirely.

**Manage Users**

This is again a subsystem solely for the use of the admin by which users would either be restricted or removed.

## Justification of the Architecture

<List down pros and cons of the architecture you have defined in the context of your system. Moreover, give a justification of why this architecture is appropriate for your system. Make sure that you also discuss how this architecture helps in implementation of non-functional requirements. >

[You must discuss how the non-functional requirements will be addressed by this architecture with appropriate reasoning.]

The Architecture model we have chosen is the MVC model. This model will be helpful in our non-functional requirements because we want to test them and modify them easily which this model can very easily allow a programmer to do. In addition , one of our main requirements is performance and reducing loading time and increasing responsiveness for websites so MVC significantly improves performance. For example , we need less time for the user to login , the smooth payment process and handling multiple requests at a single time.

**Pros:**

* MVC is helpful design pattern when planning development
* Removes unnecessary dependencies
* Removes unnecessary dependencies [Repeated]
* MVC makes model classes reusable without modification
* Extendable code
* Easier to maintain or modify
* Each part can be tested independently (Model, view, controller)

**Cons:**

* MVC pattern can be hard to understand due to the complexity and updates
* MVC must have strict rules over methods (appropriate reactions from Controller)
* Developers need to know multiple technologies in order to understand and use MVC.

# Risk Management

## Potential Risks and Mitigation Strategies

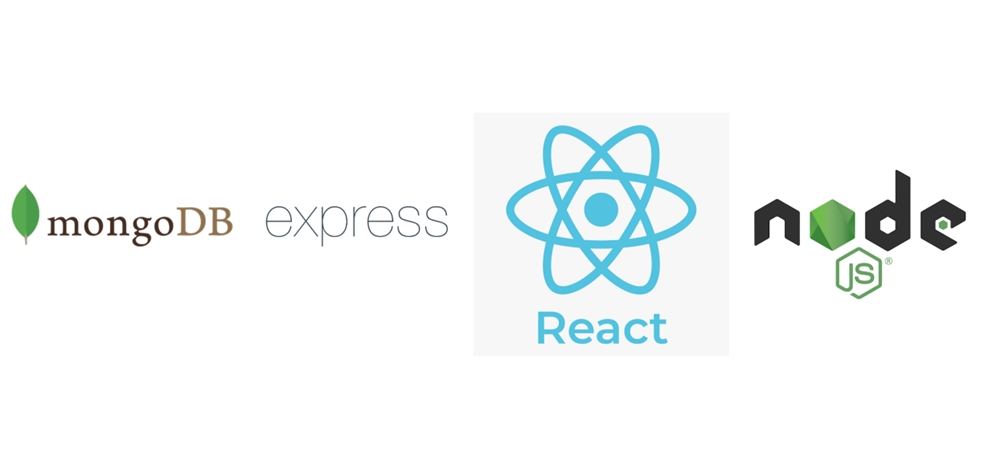
<List down top 10 potential risks and their mitigation strategies>

|  |  |  |
| --- | --- | --- |
| **Sr.** | **Risk Description** | **Mitigation Strategy** |
|  | Requirement Changes | Since we will be developing our application in sprints, and following agile methodology, it would be easier to manage requirement changes as agile methodology is flexible to changes. |
|  | Size underestimate | Develop the application over a range of size estimation and would keep a buffer before setting the size range. This would ensure that our system does not crash within the specified range. |
|  | Product Competition | Though there are competitors out in the market, we are introducing certain features which would set our application apart. For example, the hold of payment by the application till the task is completed is new in the market which would ensure a safe place for our application.Beside this we would provide better customer service along with competitive prices, to deal with the competitors. |
| 4. | Cost Estimation risk | We would go through a rigorous requirement engineering process that would set our cost boundaries. Similarly, we would get a detailed analysis of our target customers before app development. This would ensure that our app matches user needs. From a development perspective, we are using free subscription development tools and have also kept alternatives if the providers remove free subscription. For example, for deployment, we have kept Heroku, Firebase, Netlify as our main platforms which we can use for deployment. |
| 5. | Specification delays | We would ensure a detailed specification analysis process. Specification of essential interfaces would be dealt by ensuring that the critical paths are completely analyzed. |
| 6. | Hardware unavailability | We would keep a stock of hardware if one fails to ensure uninterruptible services. We would also ensure a backup of hardware services from various platforms like AWS etc. |
| 7. | Code Issues | We would ensure frequent testing and using best coding practices for example making the code modular. |
| 8. | Lower productivity | Ensure effective communication between the group mates to discuss application’s details and its problems. |
| 9. | External risks | We would ensure that our team is well informed regarding the software development laws and current events. This would help us respond quickly to external risks as they arise. |
| 10. | End-user engagement | We would ensure that our application has good affordance and visibility by following the material design. This would give customers a better user experience and engagement. |

# Tools and Technologies

<List down tools and technologies that you plan to use for development and deployment. Make sure that you mention name and version of the tools.>

We'll be using the MERN (MongoDB, ExpressJS, ReactJS, NodeJS) stack for this project's development. The MERN stack is among the most widely used technology stacks for creating web applications.



Prior to choosing this particular technological stack, several factors were taken into account.

The following is a list of some of the causes:

* Because MERN Stack only supports one language, the team is better able to work together and support one another.
* ReactJS renders and performs UI elements more quickly than other UI frameworks and libraries.
* The community has a lot of support for Javascript because it is one of the most widely used languages.
* Supports the Model View Controller protocol, which enhances the development phase's efficiency.
* The MERN stack has built-in testing libraries, which provide testing for the application.
* There are various libraries available, which makes it easier to implement many functions.
* Asynchronous programming is supported by JavaScript, which enhances performance.

MongoDB:

Using a horizontal scale-out architecture, MongoDB is an open-source document database. People use this database because it has a quick technique for retrieving data and can hold a lot of data.

ExpressJS:

ExpressJS is a JavaScript web application framework that is available as open source. ExpressJS makes it simple and rapid to construct APIs. It helps organize web applications on the server side into a more structured MVC design and is lightweight. We can effectively route our traffic using ExpressJS, and we can also build middleware to handle requests for traffic in the right way.

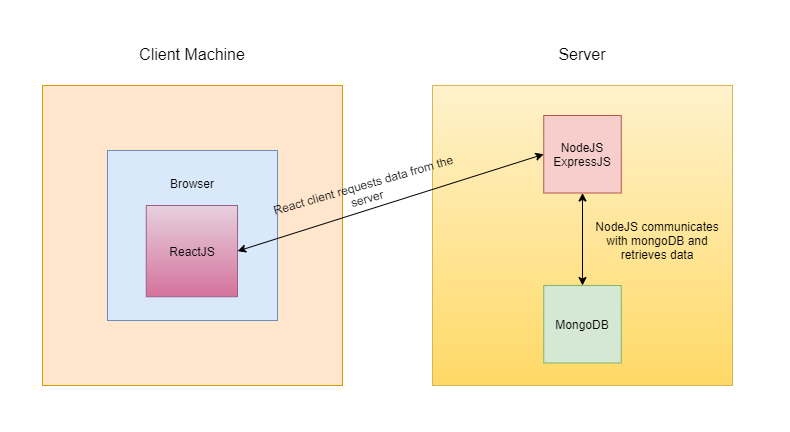
ReactJS:

For building interactive user interfaces, ReactJS is a free and open source JavaScript front end library. Facebook currently looks after it. ReactJS makes it simpler to create the straightforward views that our online application needs. Furthermore, since each component is capable of maintaining its own state, we may add new code without having it impact existing code.

NodeJS:

Performance, efficacy, and the speed at which your online application loads are all factors that a developer must consider when choosing coding languages or frameworks. As we all know, JavaScript is the most popular programming language for creating client-side web applications. Since NodeJS was introduced, JavaScript can now also be used to create server-side apps. Our application's effectiveness and efficiency are two advantages. An open source run-time javascript environment called NodeJS is based on Chrome's V8 engine. It is advantageous because it is incredibly rapid and effective, event-driven, and provides non-blocking I/O. It can complete multiple tasks at once without hindering or interfering with other operations. As a result, the performance of the websites is greatly enhanced.

This figure is only for illustration purposes:



Deployment:

So for using a free hosting service we will deploy it using Infinity Free hosting services. There are some limited good features offered by the website in their free package such as :

* Free subdomains
* 5 GB disk space
* Unlimited bandwidth
* 400 MySQL databases
* Knowledge base support

We will also try to use Hostinger which is a paid hosting website but with a minal fee of approx $3 per month we are able to get a lot of features such as **100 GB** SSD Storage , **Free** Weekly Backups , **Unlimited** Free SSL , **Free** Domain , **Unmetered** traffic (Unlimited GB) , **Free** Email which is very useful and gives a much faster results.

// we can also consider aws.

Our web application will be hosted by Amazon Web Services. AWS (Amazon Web Services) is a division of Amazon that offers a safe cloud computing platform with computing capacity, database storage, content distribution, and other features. We chose it for the reasons listed below:

User-Friendly:

Amazon Web Services offers a user-friendly interface called the AWS Management Console. Utilization of the platform is comparatively easy owing to the company's well-documented web services APIs.

Flexible:

You can select the operating system, programming language, web application platform, database, and other services you need through AWS. Any service or piece of software can be easily loaded into a virtual ecosystem using the platform.

Security:

AWS has an end-to-end strategy that incorporates physical, operational, and software measures to secure and harden its infrastructure.

Scalability:

AWS offers scalability through capabilities like elastic load balancing and auto scaling, which may be utilized to scale up or down an application in response to demand. Furthermore, you have access to storage and resources anytime you need them because it is supported by Amazon's extensive infrastructure.

# Hardware Requirements

<List down the hardware requirements. This should include requirements for both development machines and deployment servers>

.   
To access the web application, the requirements are any desktop computer or mobile device with at least 2GB RAM, at least one processor consisting of a 1.6GHz CPU, and a basic internet connection. A Screen resolution of 1024x768 or higher is also recommended.

The development machines require at least 2GB RAM to complete any complex requests, and 1.6GHz or faster processor.

For the deployment servers to function efficiently, the hardware requirements include a machine with at least 4GB RAM, at least two processors consisting of 1.6GHz CPUs each, and a stable internet connection.

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Farva Talib | tools and technologies |
| Salman Masood | Risk Management |
| Muhammad Bilal Shahid | Introduction, Architecture Diagram, Architecture Description |
| Muhammad Umair Mohsin | Architecture Diagram, Justification of the Architecture |
| Muhammad Affan Ashraf | Hardware Requirements |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| Tools and Technology | Muhammad Bilal Shahid |
| Introduction and architecture diagram | Salman Masood |
| Risk Management | Muhammad Affan Ashraf |
| Hardware Requirements | Muhammad Umair Mohsin |