**Colleasy (Collaborations Made Easy)**

**SPROJ Report**



**Farva Talib 23100268**

**Muhammad Affan Ashraf 23100016**

**Muhammad Bilal Shahid 23100091**

**Muhammad Umair Mohsin 23100233**

**Salman Masood 23100254**

**Advisor: Waqar Ahmad  
School of Science and Engineering  
Lahore University of Management Sciences  
Submission Date April 21, 2023**

**Acknowledgement and Dedication**

**Certificate**

I certify that the senior project titled “**Add project title here**” was completed under my supervision by the following students:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

and the project deliverables meet the requirements of the program.

------------------------------------- Date:

**Advisor (Signature)**

------------------------------------- Date:

**Co-advisor (if any)**

# Table of Contents

[**Table of Contents 4**](#)

[**List of Figures 6**](#)

[**1. Introduction 7**](#)

[**a. Introduction 7**](#)

[**b. Objective and Scope 7**](#)

[**c. Development Methodology 8**](#)

[**d. Contributions 8**](#)

[**2. System Requirements 9**](#)

[**a. System Actors 9**](#)

[**b. Functional Requirements 10**](#)

[**c. Non-functional Requirements 10**](#)

[**3. System Architecture 12**](#)

[a. Architecture Diagram 12](#_gjdgxs)

[b. Architecture Description 14](#_30j0zll)

[c. Justification of the Architecture 16](#_1fob9te)

[**d. Tools and Technologies 16**](#)

[**4. Requirements Specifications 20**](#)

[**a. Use Cases**](#)  22

[b. Class Diagram 32](#_2et92p0)

[c. Sequence Diagrams 34](#_tyjcwt)

[**5. Software Development Methodology and Plan 47**](#)

[a. Software Process Selection 47](#_3dy6vkm)

[b. Gantt Chart 48](#_1t3h5sf)

[**6. Database Design and Web Services 51**](#)

[**a. Database Design 51**](#)

[**b. API Specification 53**](#)

[**7. System User Interface 54**](#)

[**8. Project Security 59**](#)

[**a. Project Threats 59**](#)

[**b. Potential Losses 59**](#)

[**c. Security Controls 60**](#)

[**d. Static and Dynamic Security Scanning Tools 61**](#)

[**9. Risk Management 62**](#)

[Potential Risks and Mitigation Strategies 63](#_4d34og8)

[**10. Testing and Evaluation. 65**](#)

[**11. Deployment Guidelines 66**](#)

[**12. Conclusion 68**](#)

[**a. Summary 68**](#)

[**b. Challenges 69**](#)

[**c. Future 69**](#)

[**13. Review checklist 70**](#_2s8eyo1)

[**14. References 71**](#)

# List of Figures

# Introduction

## Introduction

Colleasy 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.

## Objective and Scope

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 **Colleasy** 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.

## Development Methodology

We used Agile Methodology for the development process of our project. Agile methodology is an iterative approach to software development that focuses on delivering working software quickly and continuously improving it based on feedback. The main aspect of an agile methodology is breaking down the development process into small, manageable units of work known as sprints. Each sprint is a short period of time, typically one to four weeks, during which we focused on completing a specific set of tasks. At the end of each sprint, we aimed to deliver a working piece of software that adds value to the project. An important aspect of Agile methodology is the focus on flexibility and adaptability. Our development process was designed such that we were able to respond quickly to feedback given by our supervisor and our peers.

## Contributions

Our app streamlines the process of social media marketing. It creates a smooth way for the clients to approach their relevant social media influencers and can save their time. For small businesses to reach their target audience and expand their consumer base, social media marketing is a crucial tool. The app can assist small businesses in saving time and costs by streamlining the social media marketing process, allowing them to compete with larger companies.

There are very less to no other apps available online for this purpose. Our app idea is very simple yet very innovative in a way that when you search for the app like these on the internet, you will hardly find any other app like this. Our app makes it very easy for the clients to approach the social media influencers and they can place their orders and have a definitive way to get their work done by the social media influencers as our app has a reliable way for the payment as well. All of this combined makes a painless way for the clients to reach their relevant people and get their work done in a trust-worthy way.

# System Requirements

This chapter contains three main sections: System Actors, Functional Requirements and Non Functional Requirements. The system actors contain the three different types of users present in our project namely Admin, Client and Influencer. Most of our use cases i.e functional requirements relate to Clients and Influencers. The Functional Requirements highlight the main functionalities that our application provides however it is not an exhaustive list. This is because we followed an iterative approach and we continued to modify the use cases in each sprint. Finally, we have the Non Functional Requirements. Like functional requirements, non functional requirements are essential to any project. We have attempted to list down quantifiable non functional requirements so that the application can mimic industry standards.

## System Actors

List down the actor names and give a 2-3 lines description of the role of each actor

| **Actor Name** | **Description** |
| --- | --- |
| Admin | The admin will be overlooking the whole application and performing functions such as ensuring efficient flow of money, approving the list of influencers and clients on board, and blocking any actor if needed. |
| Client | The client will log in and see the list of influencers according to their needs. Clients will be able to apply filters for a narrower search, see their contact details, and pay and rate them after task completion. |
| Influencer | The influencer will log in and see his/her messages to see if any client has contacted them. He/She will then be able to either accept or reject the client’s offer. The influencer will be able to rate the experience with the client after payment. |
|  |  |

## Functional Requirements

List down system requirements. You may group requirements according to actors or modules

| **Requirements** | |
| --- | --- |
| **Sr#** | **Requirement** |
| 1 | As an Admin, I should be able to login to my account if my account is already created. |
| 2 | As an Admin, I should be able to control the payment method and its flow between the application users. |
| 3 | As an Admin, provide a list of influencers based on category. |
| 4 | As a customer, I should be able to login to my account if it is already created. |
| 5 | As a customer, I should be able to sign up to create a new account. |
| 6 | As a customer, I should narrow my search for influencers using filters. |
| 7 | As a customer, I should be able to see a list of influencers based on the entered filter. |
| 8 | As a customer, I should be able to view contact details of a particular influencer. |
| 9 | As a customer, I should be able to view my profile which contains a list of all my projects (successful and ongoing) and details associated with them. |
| 10 | As a customer, I should be able to rate influencers after task completion. |
| 11 | As a customer, I should be able to make payment to the influencer. |
| 12 | As an influencer, I should be able to login to the account if already created. |
| 13 | As an influencer, I should be able to sign up to create a new account. |
| 14 | As an influencer, I should be able to check my inbox containing the messages from customers. |
| 15 | As an influencer, I should be able to accept a request for marketing and send confirmation to customers. |
| 16 | As an influencer, I should be able to reject a request. |
| 17 | As an influencer, I should be able to rate the customer. |

## Non-functional Requirements

Although functionality and design are important for a successful user experience, a user notices a website's performance immediately. Studies show that 47% of people don't visit websites with load times of more than two seconds. A single second can cost an e-commerce website millions of dollars per year because it can reduce conversion rates by 7%. A website's speed has an impact on its SEO rating, therefore one that loads quickly has advantages. Thus, it is clear that performance is important while trying to assure a good user experience.

(<https://www.browserstack.com/guide/why-website-speed-is-important>)

| **Sr#** | **Requirements** |
| --- | --- |
| 1 | At no point during operation should the system use more memory than 1 GB. |
| 2 | No more than three failures should occur in the system per 24-hour period. In the event of a failure, the system should resume normal operations within five minutes. |
| 3 | It shouldn't take more than 5 seconds for a user to log in. |
| 4 | Authentication should take no more than 2 or 3 seconds. |
| 5 | Searching for influencers should not take more than 3 seconds. |
| 6 | Filtering influencers by area of expertise shouldn't take longer than 2 seconds. |
| 7 | Within two or three seconds, messaging applications should deliver texts. |
| 8 | System should be able to handle at least 10,000 users at once without crashing. (This is our initial goal which can be modified once our app gets more users.) |
| 9 | The execution of smooth payment should not take more than a minute. |
| 10 | It should not take more than a second to rate clients and influencers. |
| 11 | It should not take more than a second for an influencer to send confirmation or rejection to the clients. |
| 12 | Our app should be available to its users 24/7. |

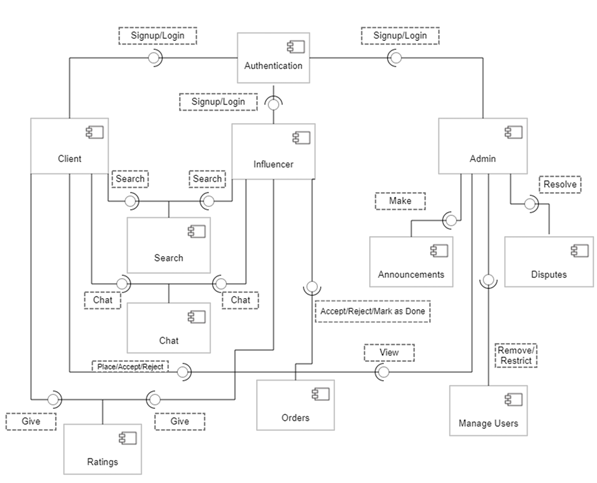
# System Architecture

This chapter describes the architecture of our app in detail. This includes the architecture diagram, its description, justification and finally comes the tools and technologies that we have used to develop our app.

## Architecture Diagram

Draw a diagram of the system architecture.

System Architecture Component Diagram



# 

# 

## Architecture Description

**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

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:**

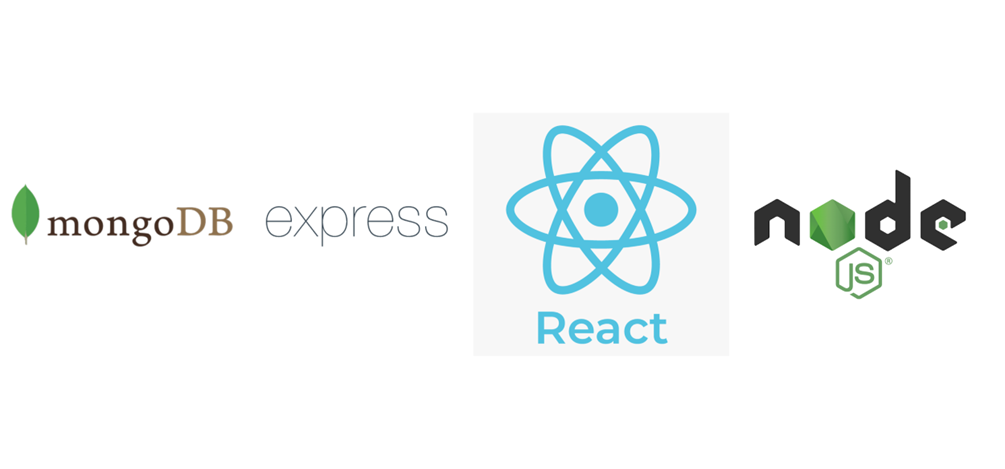
* MVC is helpful design pattern when planning development
* Removes unnecessary dependencies
* MVC makes model classes reusable without modification
* Extendable code
* Easier to maintain or modify
* Each part can be tested independently (Model, view, controller)

**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.

## Tools and Technologies

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.

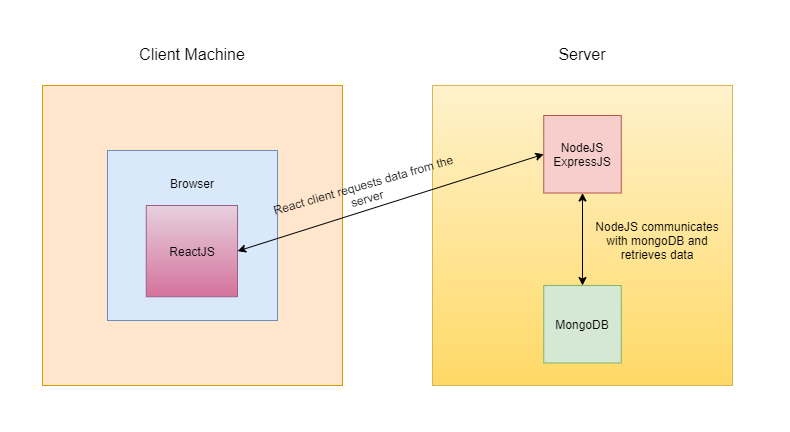
ReactJS:

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:



We used Heroku for the deployment of our application. Heroku enables programmers to easily and quickly deploy an application to a web server. Additionally, it offers a large number of plugins that you can include in your application. You can always move more quickly with a PaaS solution than you can with a VPS solution where you have to configure everything from scratch.

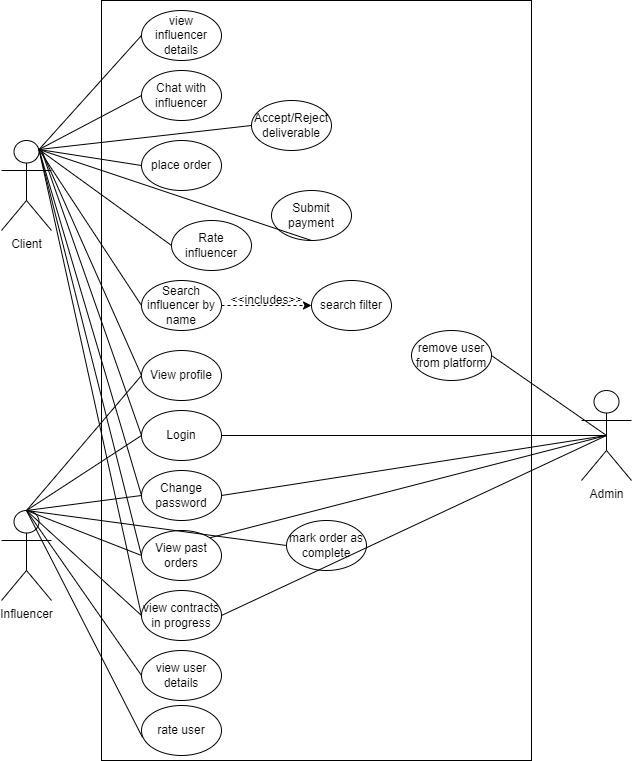
# 

# Requirements Specifications

This chapter describes the requirements of our app in detail. We have listed all the use cases for our app, their class diagram and their sequence diagrams. This gives the overall idea of the implementation of all the features in our app.

## Use Cases

Draw use case diagrams of your system using standard UML notation. Moreover, give description of 10 core use cases of your system



**Use case Table**

| **Primary Actor** | **Associated Use Cases** |
| --- | --- |
| Client | 1. Login 2. Signup 3. Change Password 4. Search an Influencer by name 5. Apply Relevant Filters 6. View profile of an influencer 7. Chat with the influencer 8. Place order 9. Submit the payment. 10. Accept/Reject the deliverable 11. View Profile 12. View contracts in progress 13. View Past Orders 14. Give a rating to the inlfuencer |
| Influencer | 1. Login 2. Signup 3. Change Password 4. View Profile 5. View Past Orders 6. View contracts in progress 7. View profile of a user 8. Accept/Reject pending contracts 9. Mark the order as completed 10. Give a rating to the client |
| Admin | Login  Signup  Change Password  View Past Orders  View Contracts in Progress  Deal with disputed contracts  Remove/Restrict Users from the platform  Send an Announcement |

### 

### 

### 

### **Login**

| **Identifier** | UC-001 |
| --- | --- |
| **Actors** | Client, Influencer, Admin |
| **Purpose** | The actor successfully logs into their account |
| **Pre-conditions** | * Actors must not already be logged in * Actors must be on the Login screen webpage * User is already registered on the platform |
| **Post-conditions** | * Actor is authenticated and logged in |
| **Typical Course Of action** | 1. Actor enter email address in the Email Address Field 2. Actor enters the password in the Password Field 3. Actor clicks the ‘Login’ button 4. The system proceeds with the authentication process and if the email address and password match, the actor is logged in 5. The use case ends |
| **Alternate Courses of Action** | * None |
| **Exception Paths** | 1. If authentication fails, then an error message is displayed 2. The user can either choose to try again or reset password by clicking the ‘forgot password’ button. |

### 

1. **SignUp**

| **Identifier** | UC-002 |
| --- | --- |
| **Actors** | Client, Influencer |
| **Purpose** | The actor successfully creates a new account |
| **Pre-conditions** | * User must not already exist in the database. Actors must be on the Signup screen webpage |
| **Post-conditions** | * Actor is authenticated and new account is created |
| **Typical Course Of action** | 1. Actor enters First Name and Last name in their respective fields. 2. Actor enters date of birth in the DOB field 3. Actor enters email address in the Email Address Field 4. Actor enters the password in the Password Field 5. Actor re-enters the password in the Re-enter Password Field 6. Actor clicks the Signup’ button 7. The system proceeds with the authentication process and if all fields match the requirements, the account is created |
| **Alternate Courses of Action** | 1. The actor can cancel the signup process at any time by clicking on the home button to be redirected to the Home Page 2. For the case of influencer, after step 5 there will be additional 3 input fields of “Niche”, “Number of followers” and “Profile Link”. |
| **Exception Paths** | 1. In step 7, if authentication fails, then an error message is displayed 2. The user can choose to try again by fixing the mistakes highlighted. |

1. **Change Password**

| **Identifier** | UC-003 |
| --- | --- |
| **Actors** | Client, Influencer, Admin |
| **Purpose** | The actor successfully resets their password |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the Profile screen webpage |
| **Post-conditions** | * Actor successfully changes the password |
| **Typical Course Of action** | 1. Actor clicks on the change password screen 2. Actors enter email address associated with their account in the Email Address Field 3. Actor enters the current password in the ‘Current Password’ Field 4. Actor enters new password in the ‘New Password Field’ 5. Actor re-enters new password in the ‘Re-Enter Password’ field 6. Actor clicks the ‘Change Password’ button 7. The system proceeds with the authentication process and if all fields are correct, the password is changed |
| **Alternate Courses of Action** | 1. The actor can abandon the process of changing password on any step of the process. |
| **Exception Paths** | 1. If authentication fails, then an error message is displayed 2. The user can choose to try again by correcting the errors highlighted. |

1. **Search Influencer by name**

| **Identifier** | UC-004 |
| --- | --- |
| **Actors** | Client |
| **Purpose** | The actor is able to search an influencer by name/hashtag from the search bar. |
| **Pre-conditions** | * Actor must be logged in * Actor must be on the HomePage of their account |
| **Post-conditions** | * Actor is able to view the relevant influencer’s profile |
| **Typical Course Of action** | 1. Actor clicks on the search bar 2. Actor types the name/hashtag of the influencer 3. The actor presses enter 4. Upon pressing enter, they will be redirected to a ‘results’ page in which relevant influencers are shown as a list based on their search query |
| **Alternate Courses of Action** | 1. The results are shown in the drop down 2. The actor clicks on the result and is redirected to the profile of the influencer |
| **Exception Paths** | 1. If the actor enters an empty string then an error message will be displayed. |

1. **View Profile of an Influencer**

| **Identifier** | UC-005 |
| --- | --- |
| **Actors** | Client |
| **Purpose** | The actor is able to view the profile of an influencer so they can place an order or for research purposes. |
| **Pre-conditions** | * Actors must be logged in * Actors must have an influencer card present in front of them either on their home screen or the results page after a search query |
| **Post-conditions** | * Actor is redirected to the profile of an influencer |
| **Typical Course Of action** | 1. Actor clicks on the influencer’s card 2. Actor is redirected to the profile page of that particular influencer. |
| **Alternate Courses of Action** | * None |
| **Exception Paths** | 1. If system crashes or times out then an error message is displayed |

1. **Place Order**

| **Identifier** | UC-006 |
| --- | --- |
| **Actors** | Client |
| **Purpose** | The actor places an order with the influencer |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the Profile screen webpage of an influencer |
| **Post-conditions** | * Actor successfully places the order |
| **Typical Course Of action** | 1. Actor clicks on the “Place Order” button 2. Actor is redirected to the “Place your Order” page 3. Actor defines the deliverable in the “Expected Deliverables” textbox 4. Actor defines their available budget for the task 5. Actor defines the due date for the task 6. Actor clicks on terms and conditions 7. Actor clicks on “Place Order” |
| **Alternate Courses of Action** | 1. The actor can abandon the process of placing the order at any step of the process by clicking on “Return to Profile” button to return to the profile of the influencer. 2. The actor can also be redirected to the Home page by clicking on the platform’s name on the top left corner. |
| **Exception Paths** | 1. If any of the mandatory fields are left empty the an error message is displayed 2. The user can choose to try again by correcting the errors highlighted. |

1. **Submit the Payment**

| **Identifier** | UC-007 |
| --- | --- |
| **Actors** | Client |
| **Purpose** | The actor successfully submits the payment for an order that they’ve placed. |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the “View contracts in progress” page. * The influencer has accepted the contract |
| **Post-conditions** | * Actor successfully submits the payment |
| **Typical Course Of action** | 1. Actor clicks on the contract for which they need to submit the payment 2. Actor is redirected towards an external website where they submit the payment (most probably Stripe) 3. Actor enters the required information and if everything is correct and required funds present in their account then the payment is made |
| **Alternate Courses of Action** | 1. The actor can abandon the process of submitting the payment on any step of the process. |
| **Exception Paths** | 1. If payment fails, then an error message is displayed 2. The user can choose to try again by correcting the errors highlighted. |

1. **Accept/Reject the Delieverable**

| **Identifier** | UC-008 |
| --- | --- |
| **Actors** | Client |
| **Purpose** | The actor accepts/rejects the deliverable |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the “View contracts in progress” page. |
| **Post-conditions** | * Actor marks the deliverable as Accepted/Rejected |
| **Typical Course Of action** | 1. Actor clicks on the contract for which the influencer has marked it completed. 2. If the task has been done and all requirements met, the actor clicks on Accept Deliverable. 3. A popup appears in which the client is asked to Rate the influencer’s work (mandatory) and also has the option to leave a review. 4. The order moves to the “View Past Orders” section and is removed from the “View Contracts in Progress” page. |
| **Alternate Courses of Action** | 1. In step 2, if the task has not been completed before the due date or if all the requirements have not been met, the actor clicks on “Reject Deliverable”. 2. Actor is redirected to a page where they have to submit proof that the requirements were not met by the influencer. 3. The admin manually verifies the case and decides which party is at fault. |
| **Exception Paths** | 1. If the actor does not give a reasoning for rejecting a deliverable then an error message is displayed. |

1. **View Profile**

| **Identifier** | UC-009 |
| --- | --- |
| **Actors** | Client, Influencer |
| **Purpose** | The actor views their own profile |
| **Pre-conditions** | * Actors must be logged in |
| **Post-conditions** | * Actor is able to view their profile |
| **Typical Course Of action** | 1. Actor clicks on the “Profile” icon on the navigation bar 2. Actor is redirected to their profile page |
| **Alternate Courses of Action** | * None |
| **Exception Paths** | * None |

1. **View Contracts in Progress**

| **Identifier** | UC-010 |
| --- | --- |
| **Actors** | Client, Influencer, Admin |
| **Purpose** | The actor is able to view the contracts in progress |
| **Pre-conditions** | * Actors must be logged in |
| **Post-conditions** | * Actor is able to view all the contracts currently ongoing |
| **Typical Course Of action** | 1. Actor clicks on “Contracts” button present in the navigation bar 2. Actor is redirected to the page where all contracts are visible that are currently in progress |
| **Alternate Courses of Action** | * None |
| **Exception Paths** | * None |

1. **View Past Orders**

| **Identifier** | UC-011 |
| --- | --- |
| **Actors** | Client, Influencer, Admin |
| **Purpose** | The actor is able to view all the past orders that they have been involved in |
| **Pre-conditions** | * Actors must be logged in |
| **Post-conditions** | * Actor is able to view all the orders placed till date |
| **Typical Course Of action** | 1. Actor clicks on “Past Orders” button present in the navigation bar 2. Actor is redirected to the page where all past orders are visible. 3. Actor can set a custom time period (optional) if they wish to view orders that occurred within that time duration. |
| **Alternate Courses of Action** | 1. In step 2, for the case of Admin, all Past Orders of the entire platform will be visible |
| **Exception Paths** | * None |

1. **View Profile of the Client**

| **Identifier** | UC-012 |
| --- | --- |
| **Actors** | Influencer |
| **Purpose** | The actor is able to view the profile of the client |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the “Contracts” page |
| **Post-conditions** | * Actor is able to view the profile of a client who has submitted a contract (not yet accepted by actor) * Actors are able to view the profile of a client whose contract they have accepted. |
| **Typical Course Of action** | 1. Actor clicks on the particular contract for which they want to view the client’s profile 2. Actor is redirected towards the contract details page 3. Actor clicks on the client’s name 4. Actor is redirected to the client’s profile page |
| **Alternate Courses of Action** | 1. In Step 1, the actor can click on the client’s name present at the bottom of the contract details and get redirected to their profile directly |
| **Exception Paths** | * None |

1. **Accept/Reject Pending Contracts**

| **Identifier** | UC-013 |
| --- | --- |
| **Actors** | Influencer |
| **Purpose** | The actor accepts/rejects the contract |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the “Contracts” webpage |
| **Post-conditions** | * If the contract is accepted, the client is notified to make the payment after which the contract would officially be in progress * If contract is rejected then the client is notified about it |
| **Typical Course Of action** | 1. Actor clicks on the contract that they want to accept/reject under the “Pending Contracts” heading. 2. Actor is redirected to the contract details page where all the relevant information regarding that job is visible 3. Actor clicks on either “Accept” or “Decline” |
| **Alternate Courses of Action** | 1. The actor can abandon the process at any stage and simply choose to do it a later time |
| **Exception Paths** | * None |

1. **Mark order as Completed**

| **Identifier** | UC-014 |
| --- | --- |
| **Actors** | Influencer |
| **Purpose** | The actor marks the order as completed |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the “Contracts” webpage |
| **Post-conditions** | * The order is marked completed and the client is notified |
| **Typical Course Of action** | 1. Actor clicks on the contract that they want to mark as completed under the “In Progress” heading. 2. Actor is redirected to the contract details page where all the relevant information regarding that job is visible 3. Actor clicks on “Mark order as Completed” |
| **Alternate Courses of Action** | 1. The actor can abandon the process at any stage and simply choose to do it a later time (provided the timer for that contract hasn’t expired already) |
| **Exception Paths** | * None |

1. **Give Rating to the Client/Influencer**

| **Identifier** | UC-015 |
| --- | --- |
| **Actors** | Client, Influencer |
| **Purpose** | The actor gives a rating to the client that they’ve just finished a contract with |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the Contract webpage * The client has accepted the deliverable * The Influencer has received the payment |
| **Post-conditions** | * Actor gives a rating to the client |
| **Typical Course Of action** | 1. Actor clicks on the “Give Rating” button present on the right side of the contract details under the “Delivered” section. 2. A modal pops up in which the actor can give a rating between 1-5 to the client 3. There is also a textbox (optional) in case the actor wants to leave a review |
| **Alternate Courses of Action** | 1. In Step 2, Actor can close the popup and abandon the process altogether. |
| **Exception Paths** | * None |

1. **Remove User from the Platform**

| **Identifier** | UC-016 |
| --- | --- |
| **Actors** | Admin |
| **Purpose** | The actor removes a user from the platform |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the Users webpage |
| **Post-conditions** | * The selected user is removed from the platform and all access is revoked. |
| **Typical Course Of action** | 1. Actor clicks on the user that they wish to remove 2. Actor is redirected towards the profile of that user 3. Actor clicks on “Remove User” button 4. Actor is asked via a popup whether they are sure about their decision 5. Actor clicks on “Yes” 6. Actor is then asked to enter their password 7. User is deleted and Actor is redirected towards the “Users” page |
| **Alternate Courses of Action** | 1. The actor can abandon the process of removing a user at any time |
| **Exception Paths** | 1. If authentication fails of the password, an error message is displayed. 2. The actor can re enter the password once. 3. If the password matches the user is deleted. 4. Otherwise the admin is restricted to delete the user by 30 minutes and an email is sent on the official email address of the admin regarding the whole procedure. |

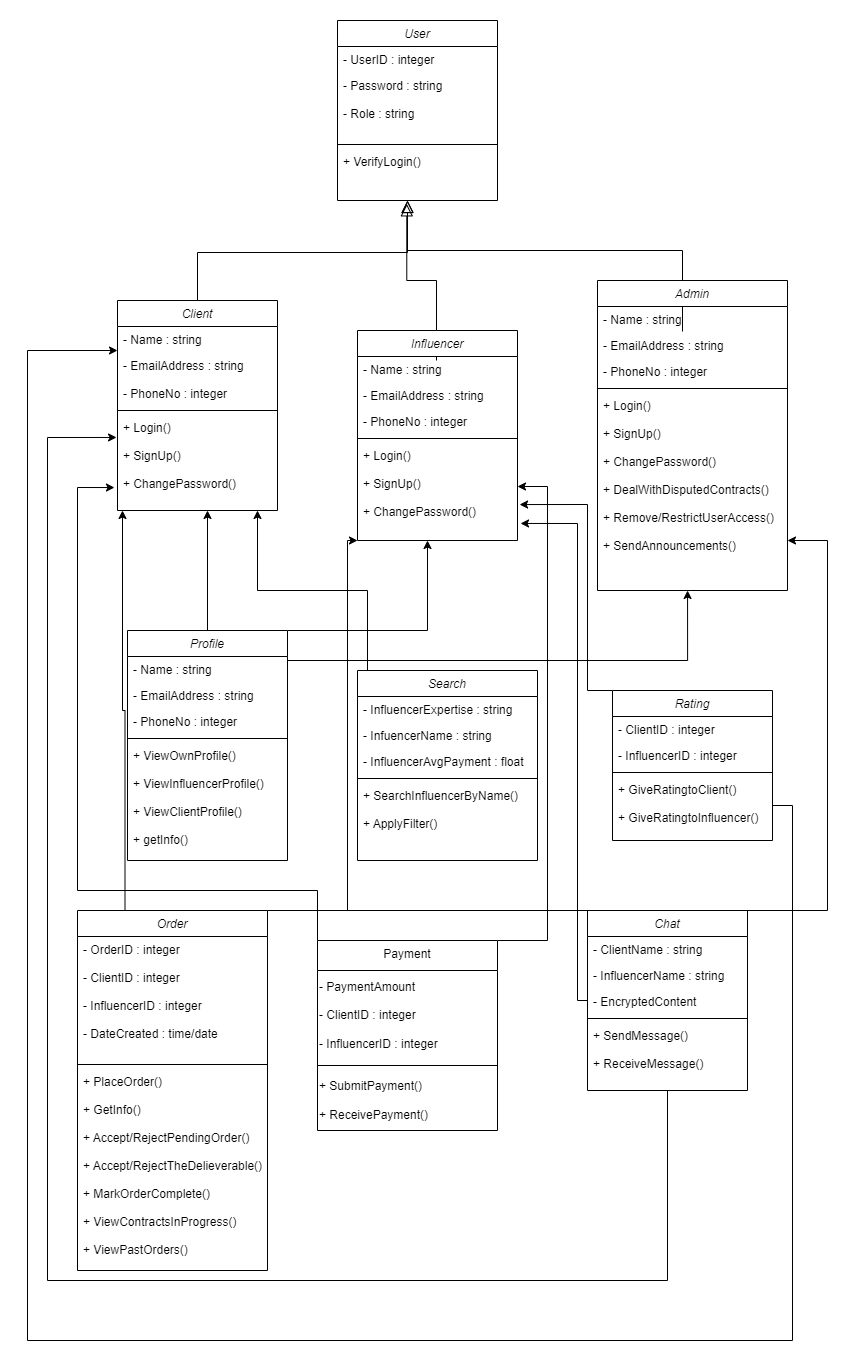
1. **Send an Announcement**

| **Identifier** | UC-017 |
| --- | --- |
| **Actors** | Admin |
| **Purpose** | The actor sends an announcement to all other actors registered on the platform |
| **Pre-conditions** | * Actors must be logged in * Actors must be on the Announcements webpage |
| **Post-conditions** | * Actor successfully sends the announcement |
| **Typical Course Of action** | 1. Actor clicks on the “New Announcement” button 2. Actor types in the new announcement 3. Actor clicks on “Send” |
| **Alternate Courses of Action** | 1. The actor can abandon the process of sending an announcement at any stage |
| **Exception Paths** | 1. If the actor submits an empty field as the new announcement then an error message is displayed 2. The actor can fix the error and proceed as they would do so in the typical course of action |

## Class Diagram

Use standard UML notation to draw the class diagram. Give brief description/purpose of each class in the class diagram. Give readable names to classes, attributes and operations.

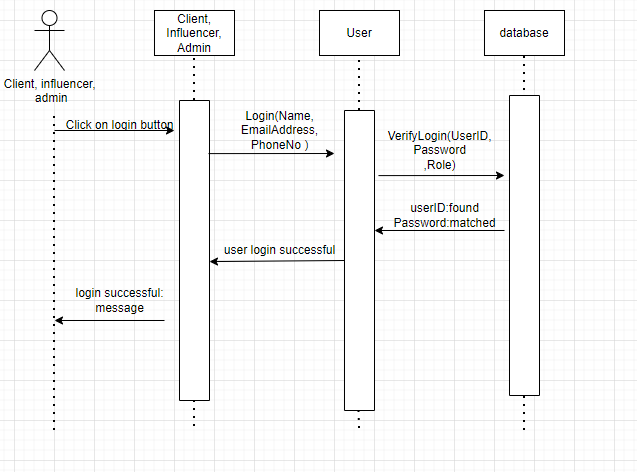
We have three main classes; client, influencer and admin. These are the main actors of our app. All three lie in the category of users. They all can login, sign up, change password, view past orders and view pending orders. Other classes include profile, search,order,rating and chat etc, And they all have different operations related to each class.



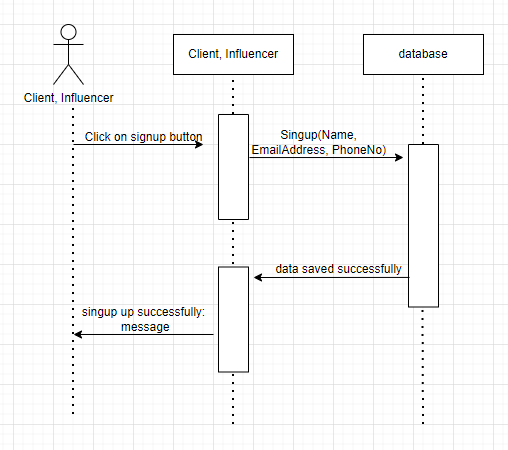
## Sequence Diagrams

Draw sequence diagrams of 10 core use cases. Draw the diagrams using standard UML notation

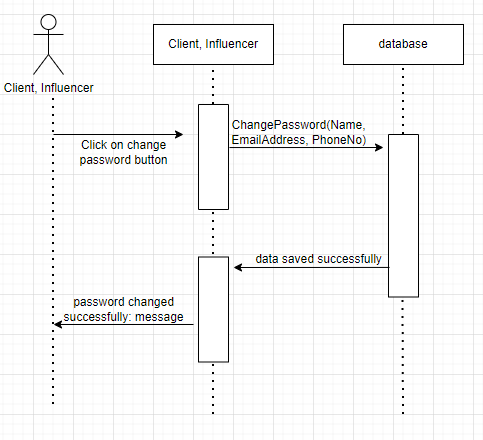
1. **Login**



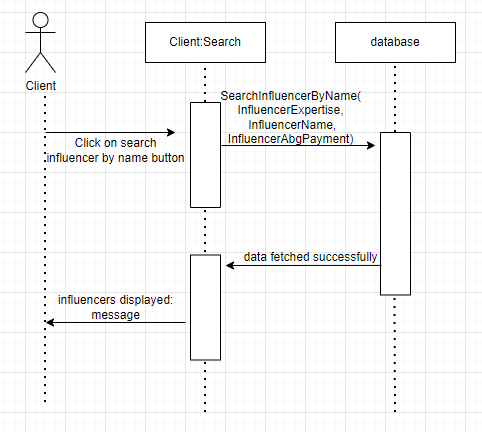
1. **Signup**



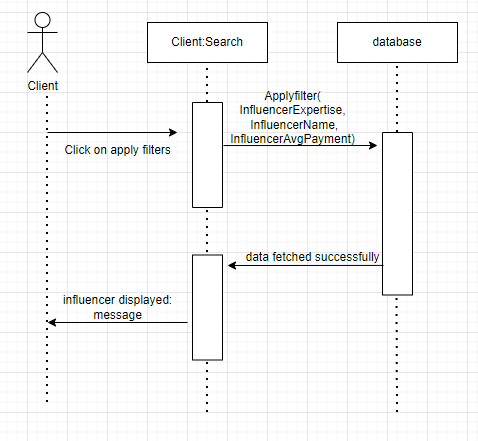
1. **Change Password**



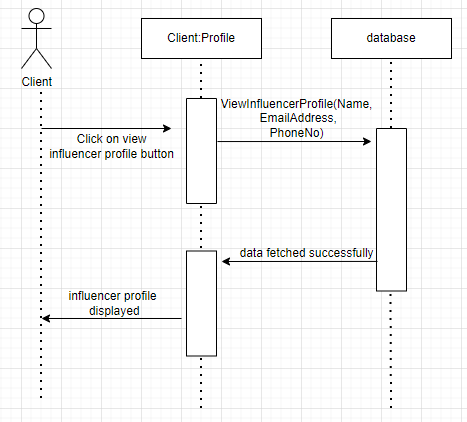
1. **Search an Influencer by name**



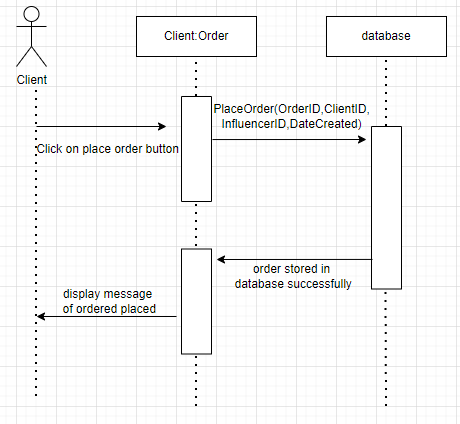
1. **Apply Relevant Filters**



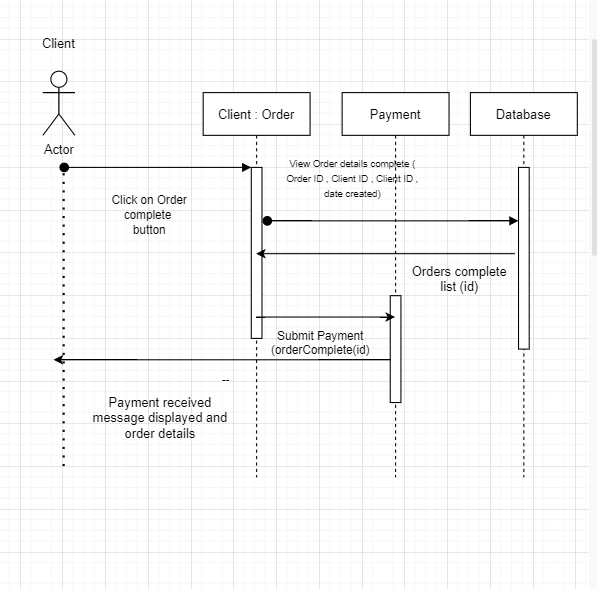
1. **View profile of an influencer**



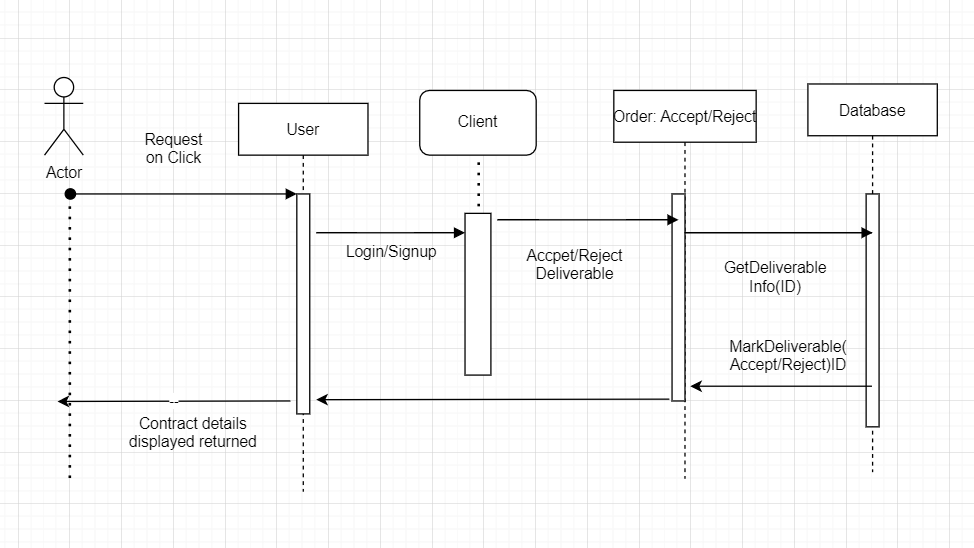
1. **Place order**



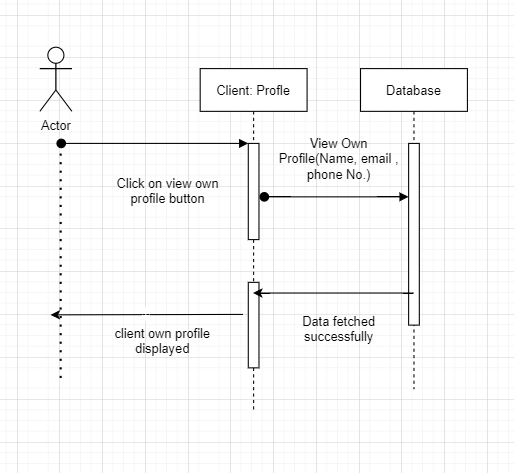
1. **Submit the payment.**



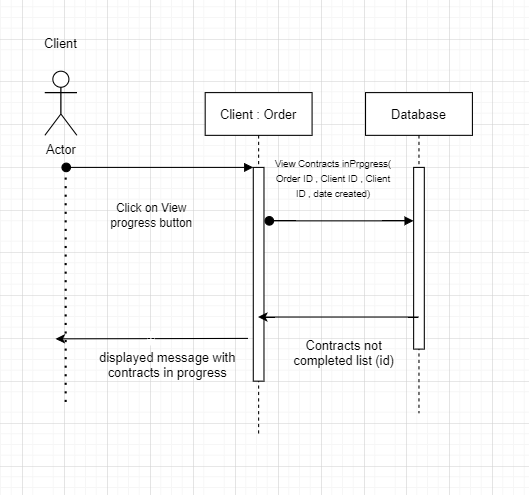
1. **Accept/Reject the deliverable**



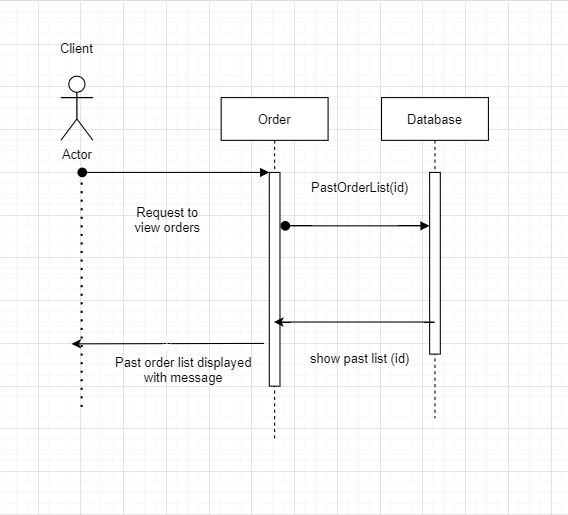
1. **View Profile**



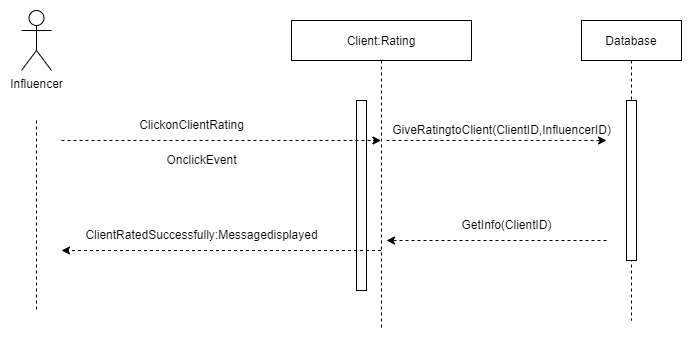
1. **View contracts in progress**



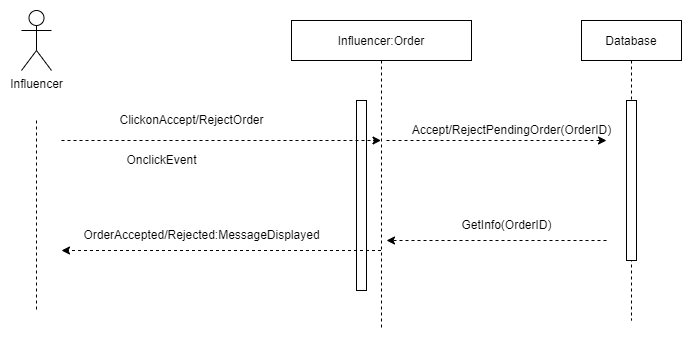
1. **View Past Orders**



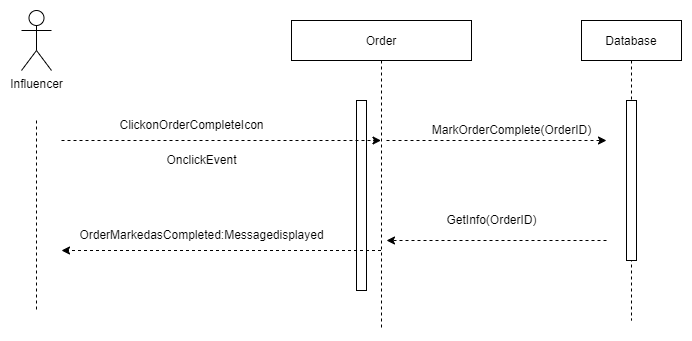
1. **Give a rating to the Client**



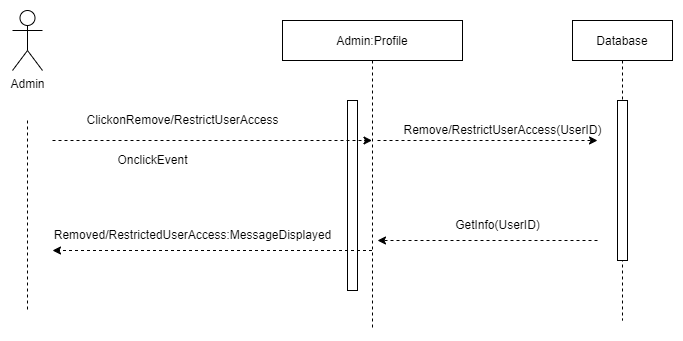
1. **Accept/reject pending contracts**



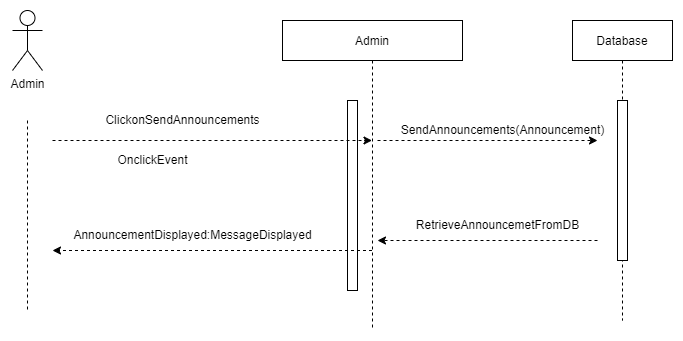
1. **Mark Order as Completed**



1. **Remove/Restrict User Access:**



1. **Send Announcement on Platform:**



# Software Development Methodology and Plan

This describes the development methodology for our app and why we have used this specific development methodology. This also includes the Gantt chart which shows the progress of our work in detail.

## Software Process Selection

1. **Waterfall Model :**

Pros :

* It's helpful for developers and people to work quickly and up to speed since this model follows in order such as technical documentation which is helpful for them to understand the objectives.
* Helps testing process made easier and transparent
* The timescales are kept in order because in the waterfall model the phase developments enforces discipline which makes each step easy to monitor.
* Helps in dealing with issues in the design phase.

Cons:

* Needs can be difficult to defined because of a structured plan. If a client wants changes to be made in any phase it has to be re-engineered to a large extent and follow the phases in steps.
* Potential lack of flexibility to cater for new development changes or requirement changes.
* Longer time for project delivery. As it takes much longer time for processes to complete in an order as compared to iterative approach methods such as Agile.

1. **Agile Model :**

Pros:

* Faster delivery of project to client. If you need a faster project completion in urgency can be completed by this method , it does not have the perfect increments but might help build the task.
* It’s Adaptable since the increments are small it's easy to modify and adapt the process of the projects according to the circumstances.
* Quickly helps in detection of problems since testing is done incrementally so if a problem occurs you can precisely solve it and fix it before the next cycle.
* It’s a transparent approach and all your workings are constantly shown.
* It’s a collaborative approach as Agile requires a lot of feedback back and forth between teams , clients etc so it helps for a collaborative environment and innovates creativity.

Cons:

* It causes some tricky paradigm shift because as some industry people use this approach but for some people however some people can’t adjust to fast paced rapid work
* There is a lack of overall cohesion as it's very easy for a continuous process to run itself and goals to get lost in details.
* There is a neglect of paperwork as well during the process. This approach often requires quick shifts from one aspect of a project to another. This may leave little time for doing the proper paperwork at each stage. Record-keeping is important, but it is often a casualty of agile working methods.

We followed an agile model for our application:

The flow of our team work and requirements of the application make us inclined towards an agile model. Since this project is based on a time constraint therefore the requirements might need some flexibility or change in future as the development proceeds. Due to the short time and pressure of developing an early working prototype within the semester, our team requires an adaptive temperament where the changes are made instantly. Therefore it has a high rate of requirement change. Moreover, as our project requires constant feedback from the instructor, a flexible approach would suit the application which can only be obtained if we follow the agile methodology. This continuous feedback would also minimize risks in the application development and would help us better meet the user needs. Since agile methodology works in sprints which requires repetitive planning and meetings, the application is developed in small portions and is then added up to the previous version, this can cause unexpected errors and defects in the program. We need to have a model that has a low potential loss due to these defects. Fortunately, the agile model facilitates the developers by reducing the loss as, for instance, if an error arises in a sprint, it is only confined to that particular sprint and not the entire program. Only that sprint needs to be corrected. Therefore, we chose agile as it helped us save time in software testing.

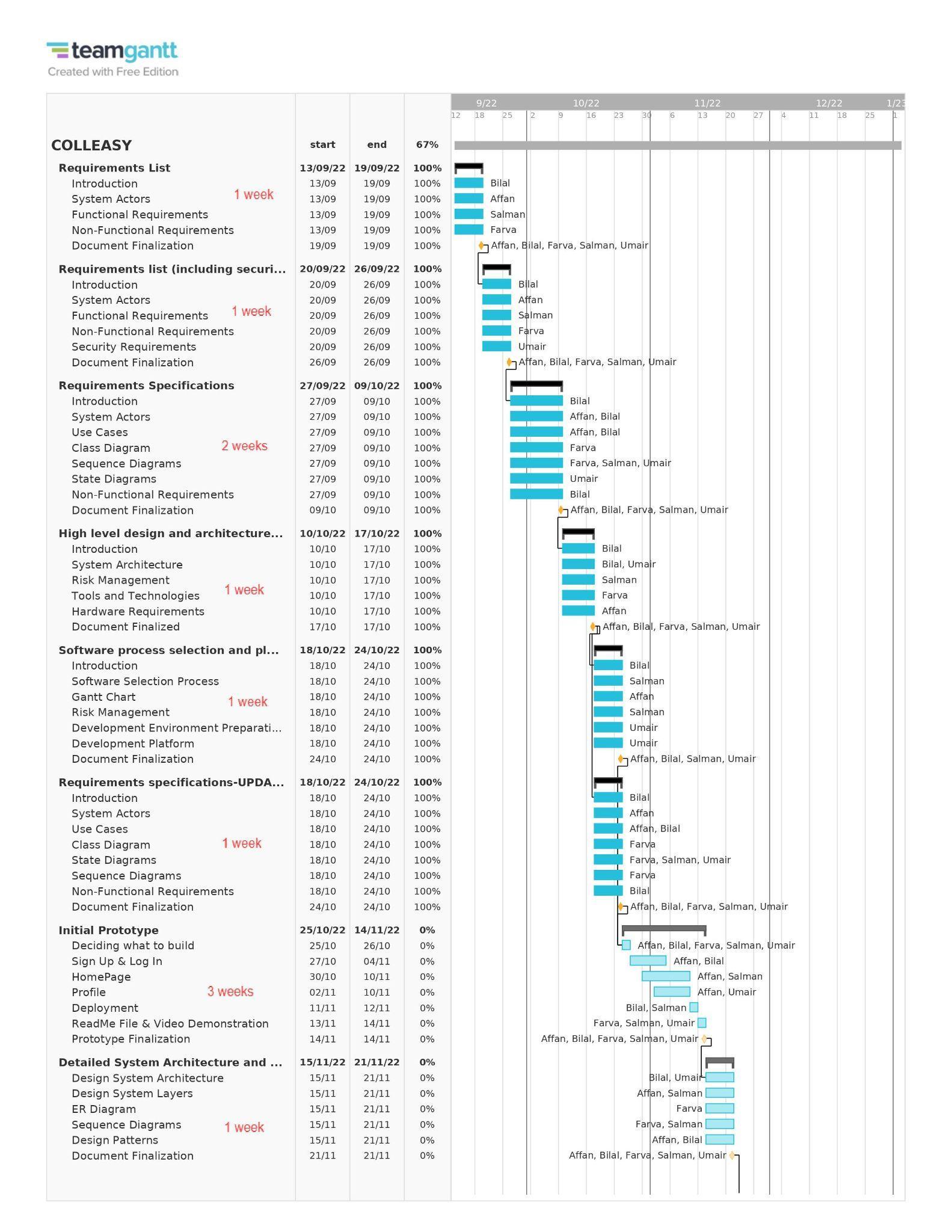
Discuss a pros and cons of waterfall and agile (scrum) processes in your own words.

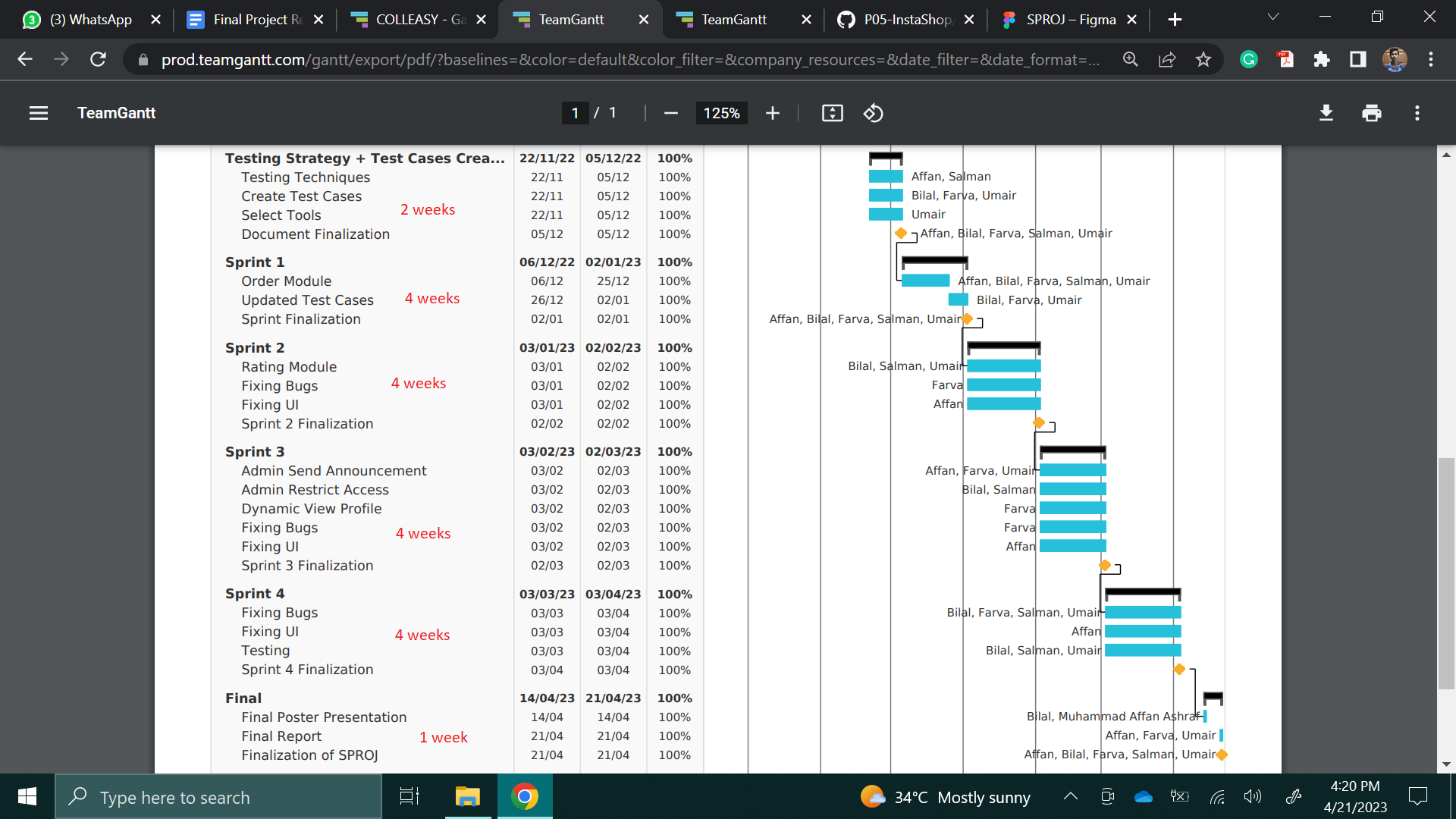
Give proper justification of the software development process that you have used for your project.

## Gantt Chart

Draw a Gantt chart that illustrates your project’s schedule. The Gantt chart should show at least the following

* Tasks (tasks should not be too small or too large)
* Duration (in weeks)
* Milestones
* Team member names who have worked on each task.





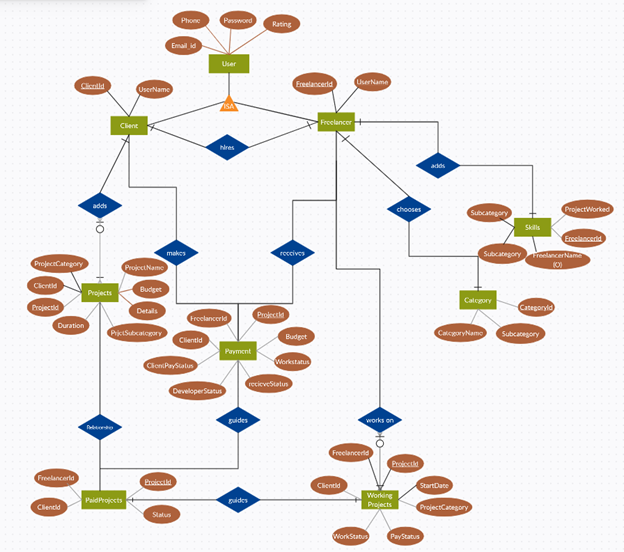
# Database Design and Web Services

This chapter describes the E/R diagram for our project and describes their details as well. Also, mentions the details of the external Web API that we have used in our app.

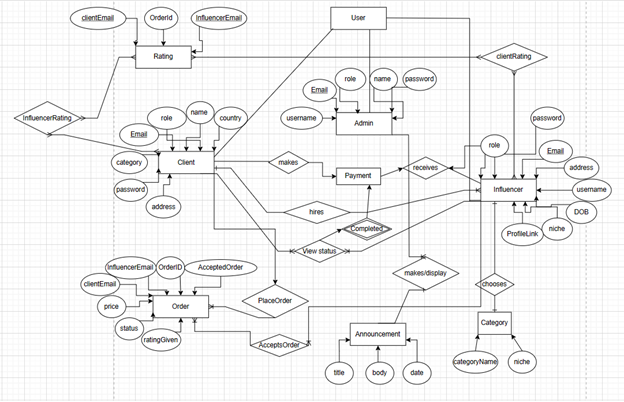
## Database Design

ER diagram for our Model :

For an overall view of the model is shown below : Freelancer is same as Influencer in our platform



Along with our Project cases and entities as well :



To start with description of ER relationships:

1. Firstly A User has a choice to be an Admin , client or an influencer when he signs up for an account and how he wants to register himself so the relation is single 1-1 as each entity is unique so a user can become one of them only.
2. Client Rating system (client-influencer Rating) is Many -to-Many because many several clients registered can give rating to any influencer for the orders/project they completed so this is the relationship.
3. Similarly Influencer Rating system (influencer- clientRating) is also Many-to-Many as well because many of the influencers who complete the order placed by clients are possible so multiple ratings of clients can be given by influencers.
4. Client and Order relationship (client-PlaceOrder) is Many -to- Many because a registered clients can place more than 1 orders against an influencer for different types of categories/niches and set the price , status , and email respectively against that particular order.
5. Admin to Announcement (One -to- Many) as an admin has the option of adding an announcement which is displayed to the users side. Which include a title and body of text shown to them.
6. Influencer- Category (One-to-One) is because an influencer has a choice of choosing one type of niche/category only.
7. Client - View Status ( One- to- Many) as clients have placed an order (multiple orders possible) it can view in the progress tab if the order is pending or completed yet or not by the influencer.
8. Influencer - View Status (One -to-Many) as influencer has received an order (multiple orders possible) it can view in the progress tab if the order is pending (if not accepted yet) or completed yet by the influencer against the OrderID.
9. Influencer - AcceptOrder( One -to-Many) relationship is there as an influencer can accept as many orders as possible according to his needs or price preference as well.

## API Specification

We have integrated Stripe API in our app.

A well-known payment processing provider called Stripe offers businesses an easy-to-use API for accepting online payments. Developers may create web applications that securely accept payments and manage transactions by integrating Stripe API with their app.

There are various advantages to using Stripe API, including:

**Security:** To protect sensitive payment information, the Stripe API complies with PCI security requirements and is PCI-compliant.

**Flexibility:** A variety of payment mechanisms, including credit cards, debit cards, digital wallets, and others, are supported by the Stripe API.

**Integration:** Adding payment processing capability to your web application is simple thanks to Stripe API's seamless integration with the MERN stack.

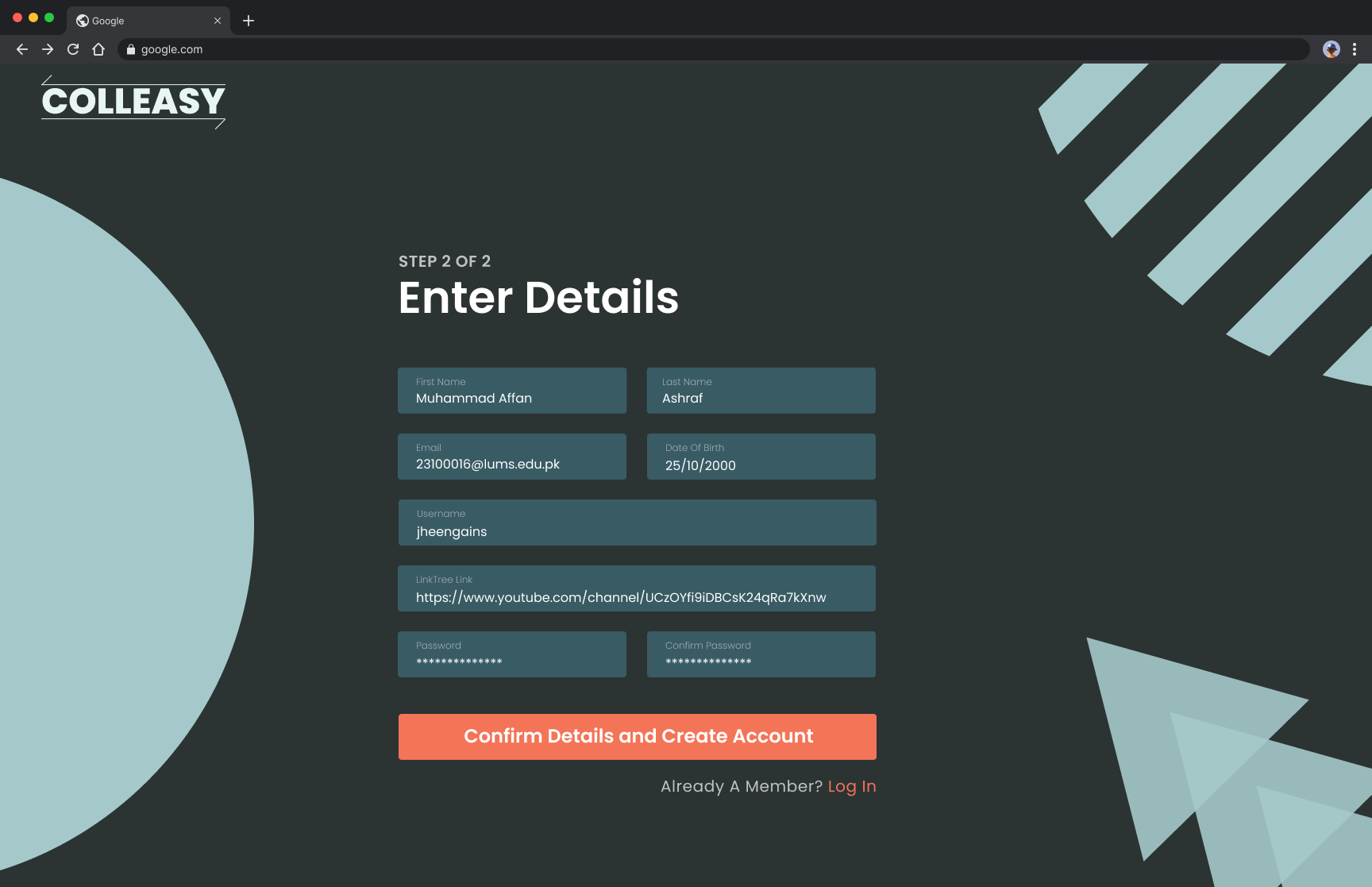
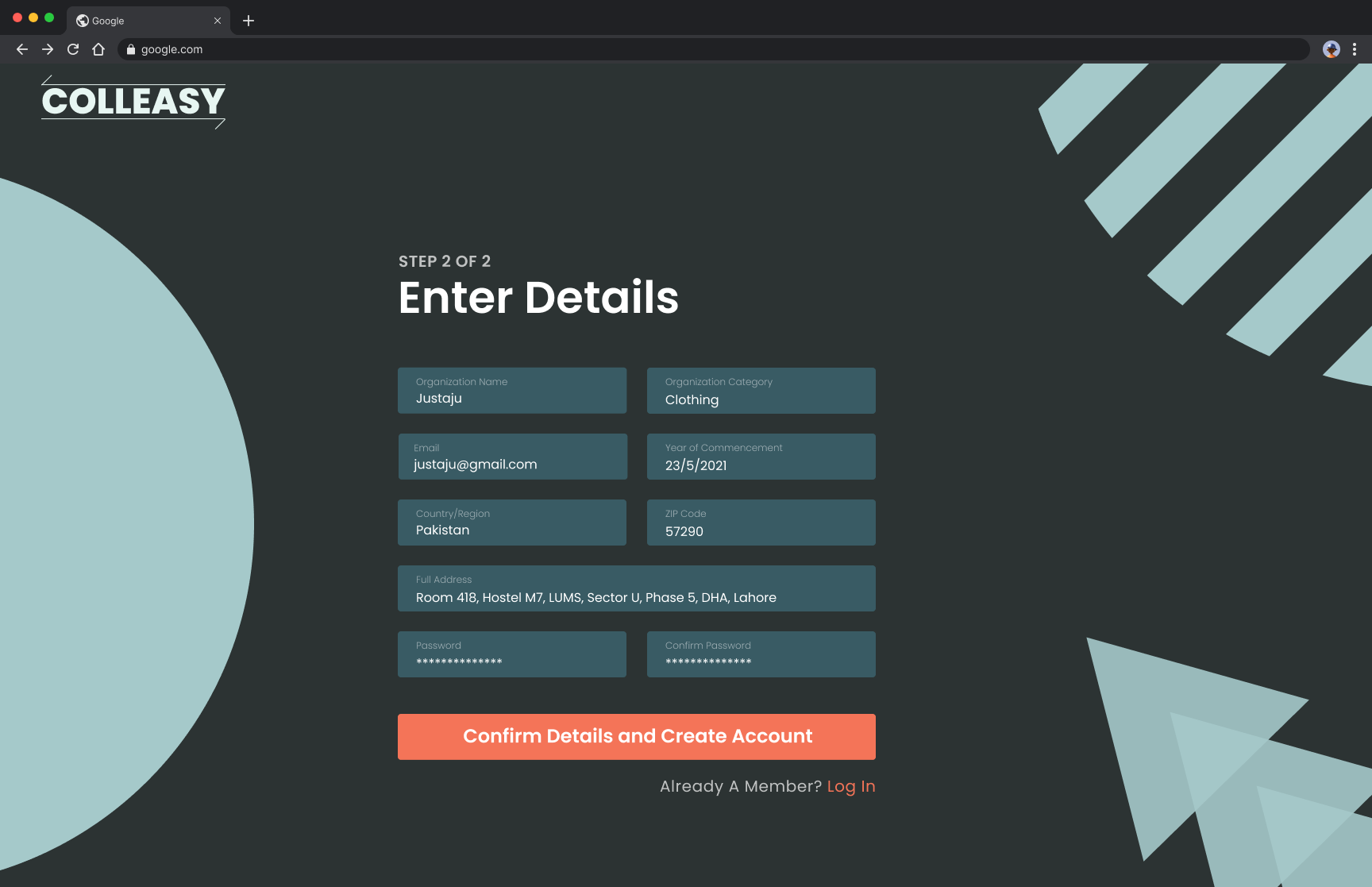
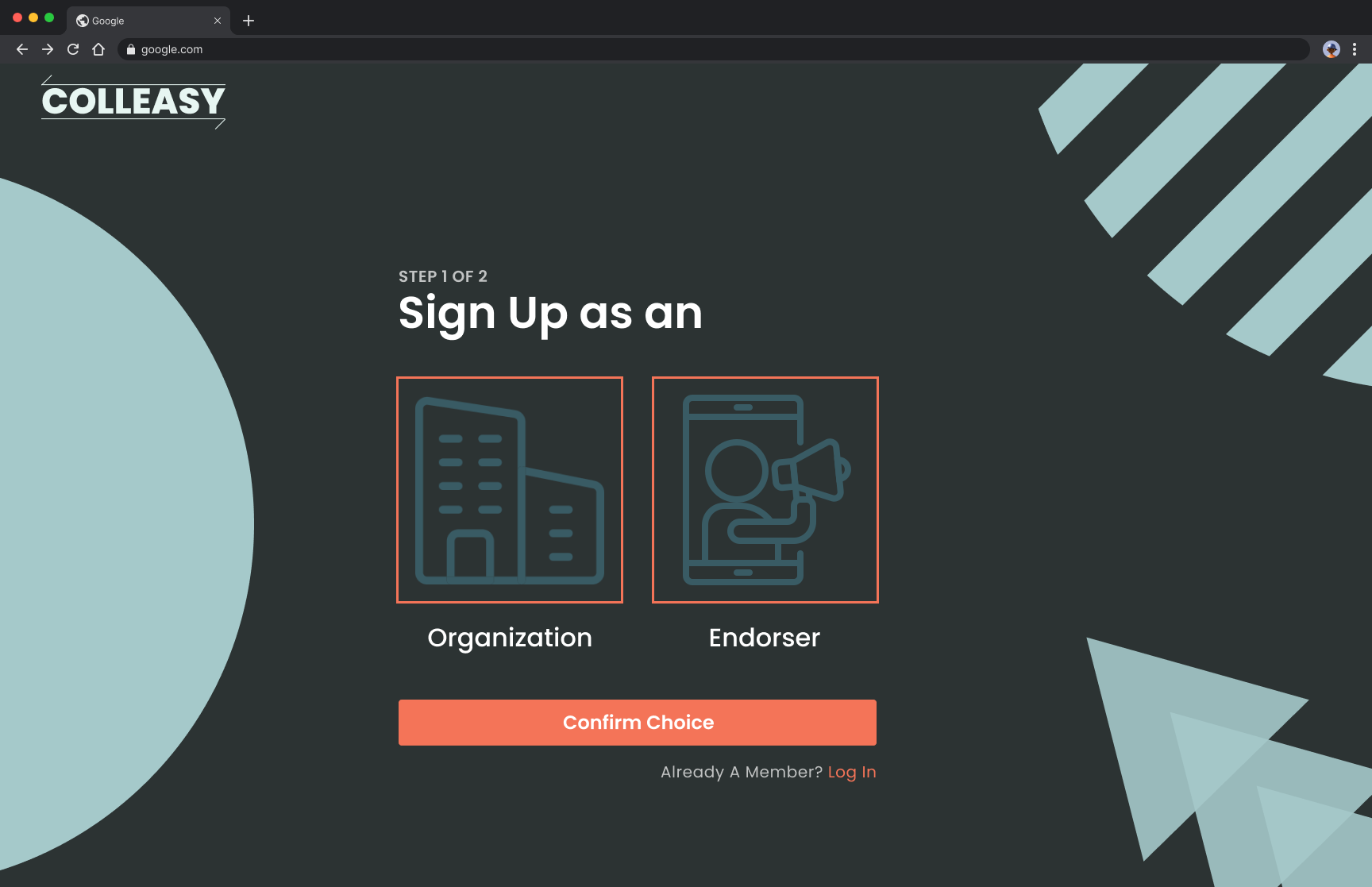
# System User Interface

The system user interface of CollEasy has been designed with a user-centric approach, keeping in mind the needs and expectations of the target audience. The design is intuitive and easy to navigate, ensuring that users can quickly and easily access the features they need. The color scheme and typography are carefully selected to create a visually appealing and cohesive look and feel. The layout is structured to provide a clear hierarchy of information and functionality, with important elements prominently displayed. User feedback has been taken into account throughout the design process to ensure that the interface is user-friendly and meets the needs of the users.

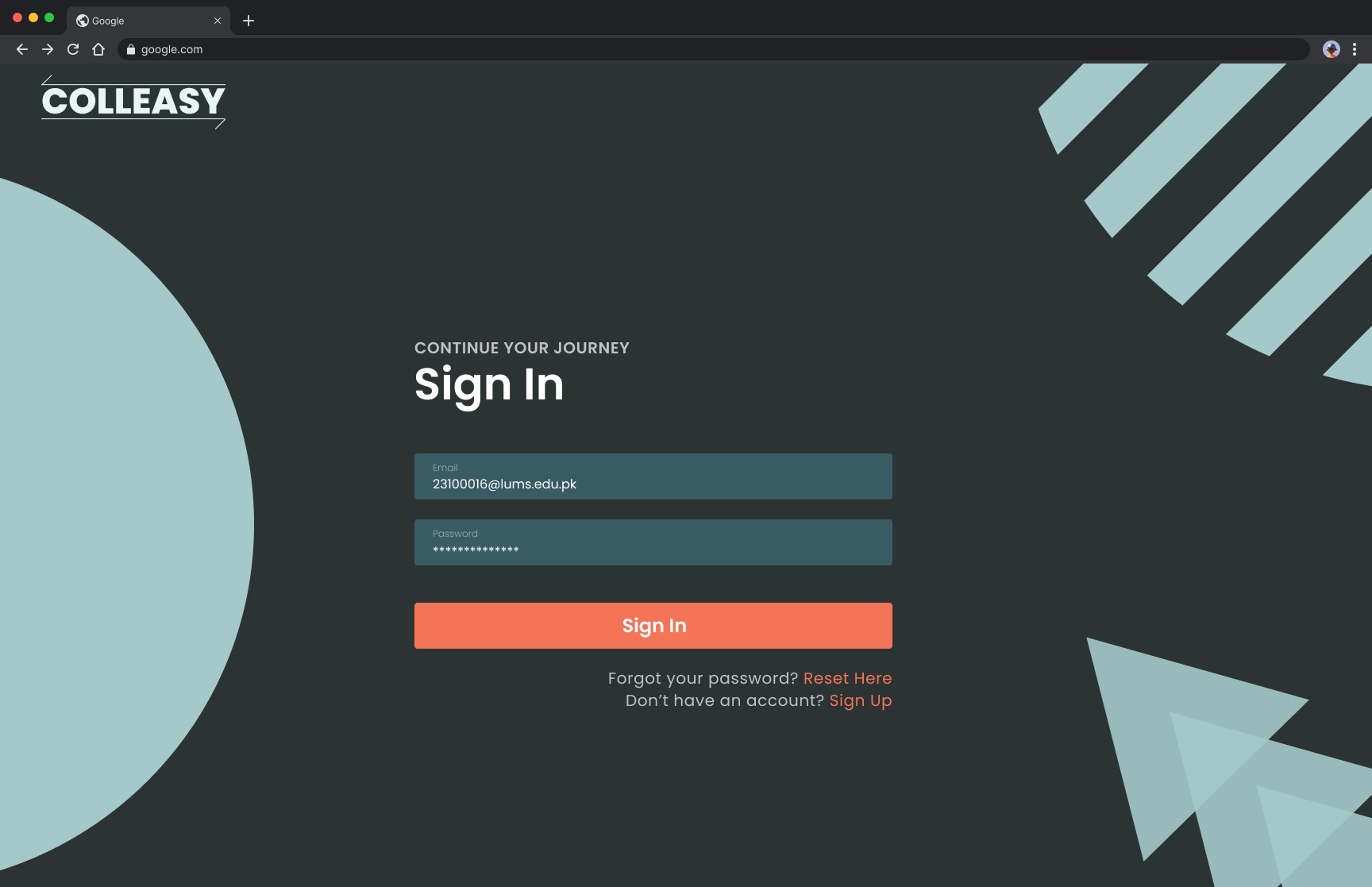
Following is the walkthrough of the application.



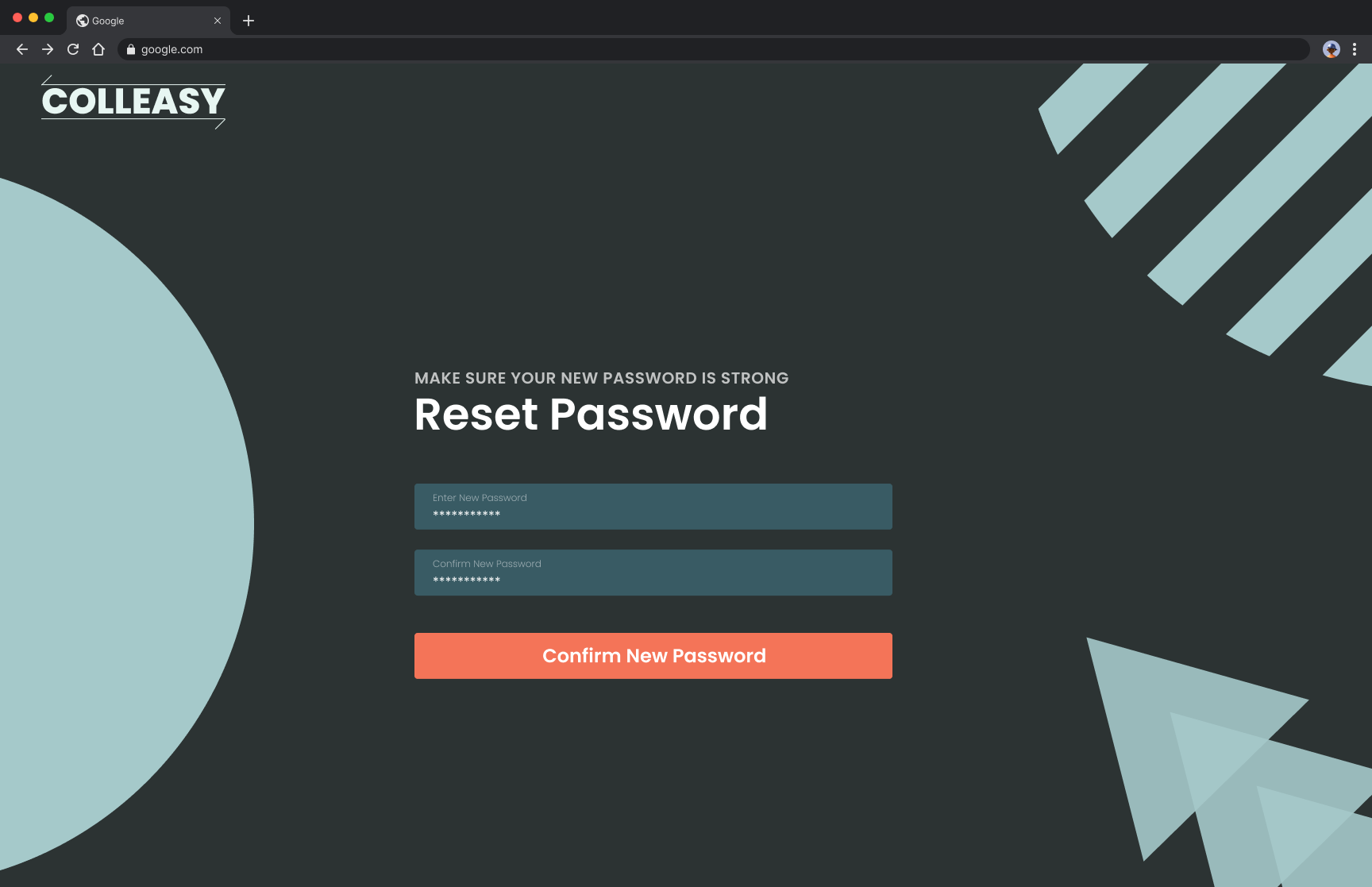
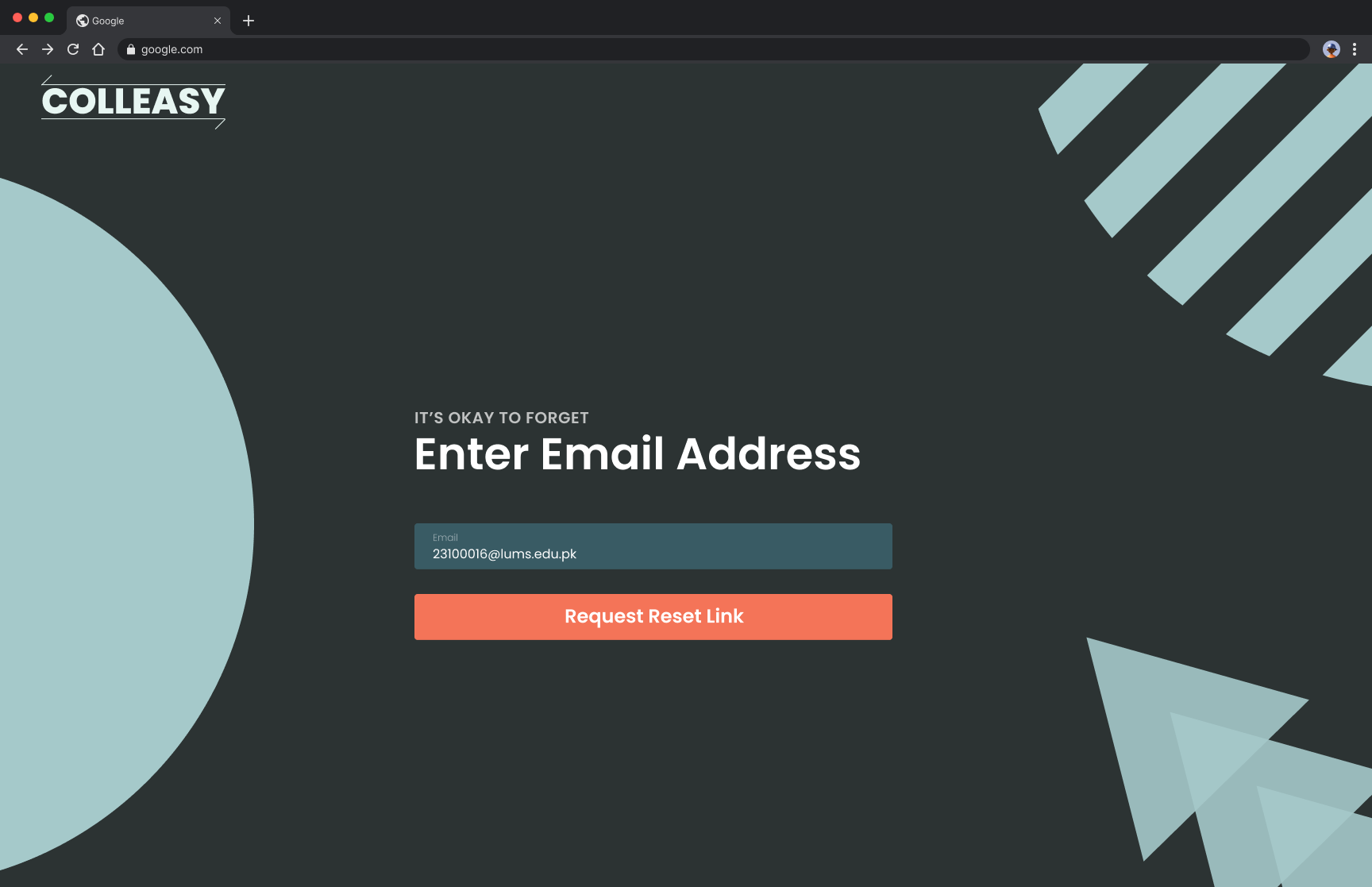
The user is directed to the landing page once the user enters the website url.



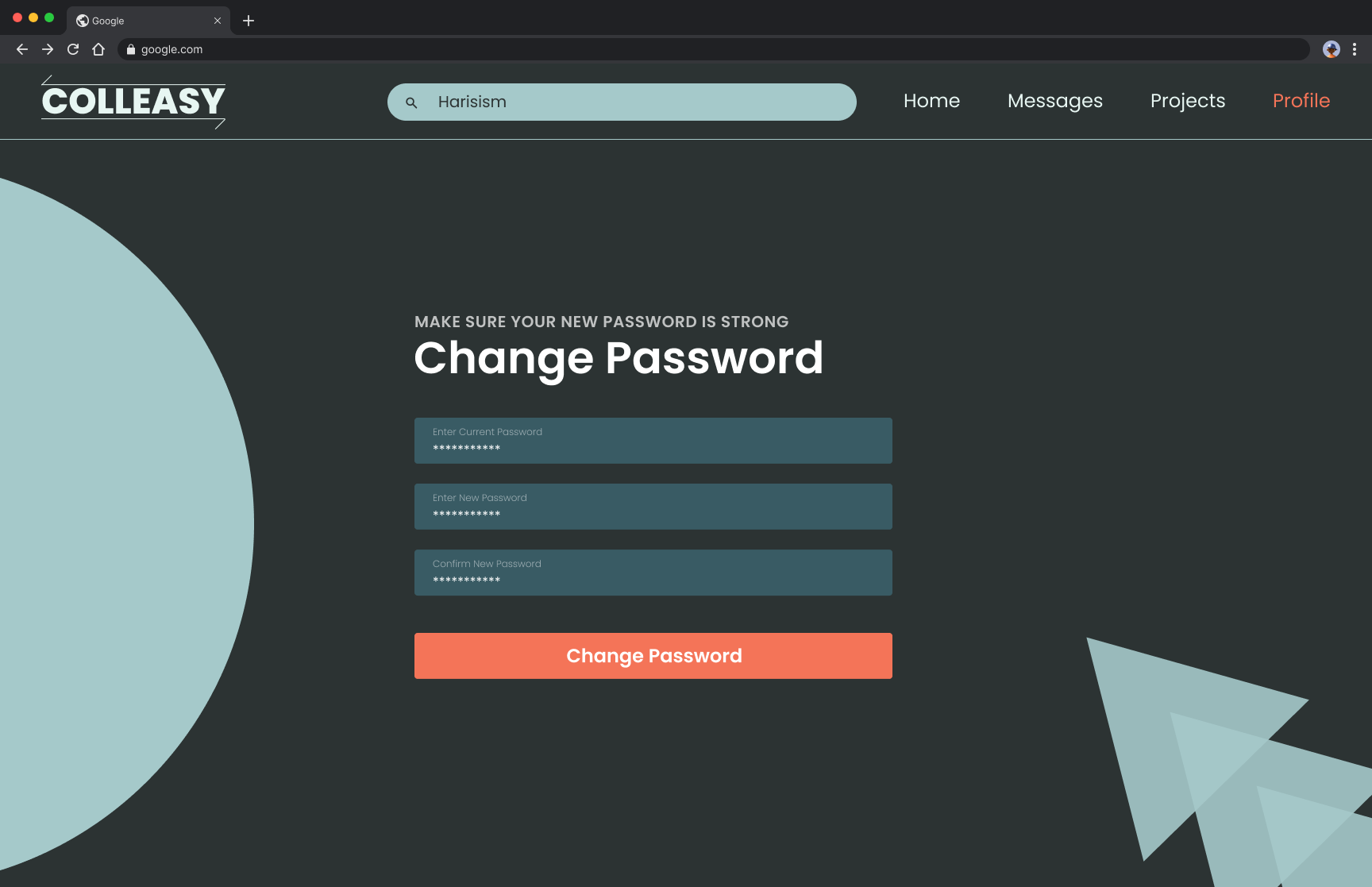
When the user clicks on signup, he is first asked to choose whether he’d like to sign up as an organization or the endorser. Based on his choice, he is navigated ot the next screen where he can enter his details to sign up and create account on CollEasy.



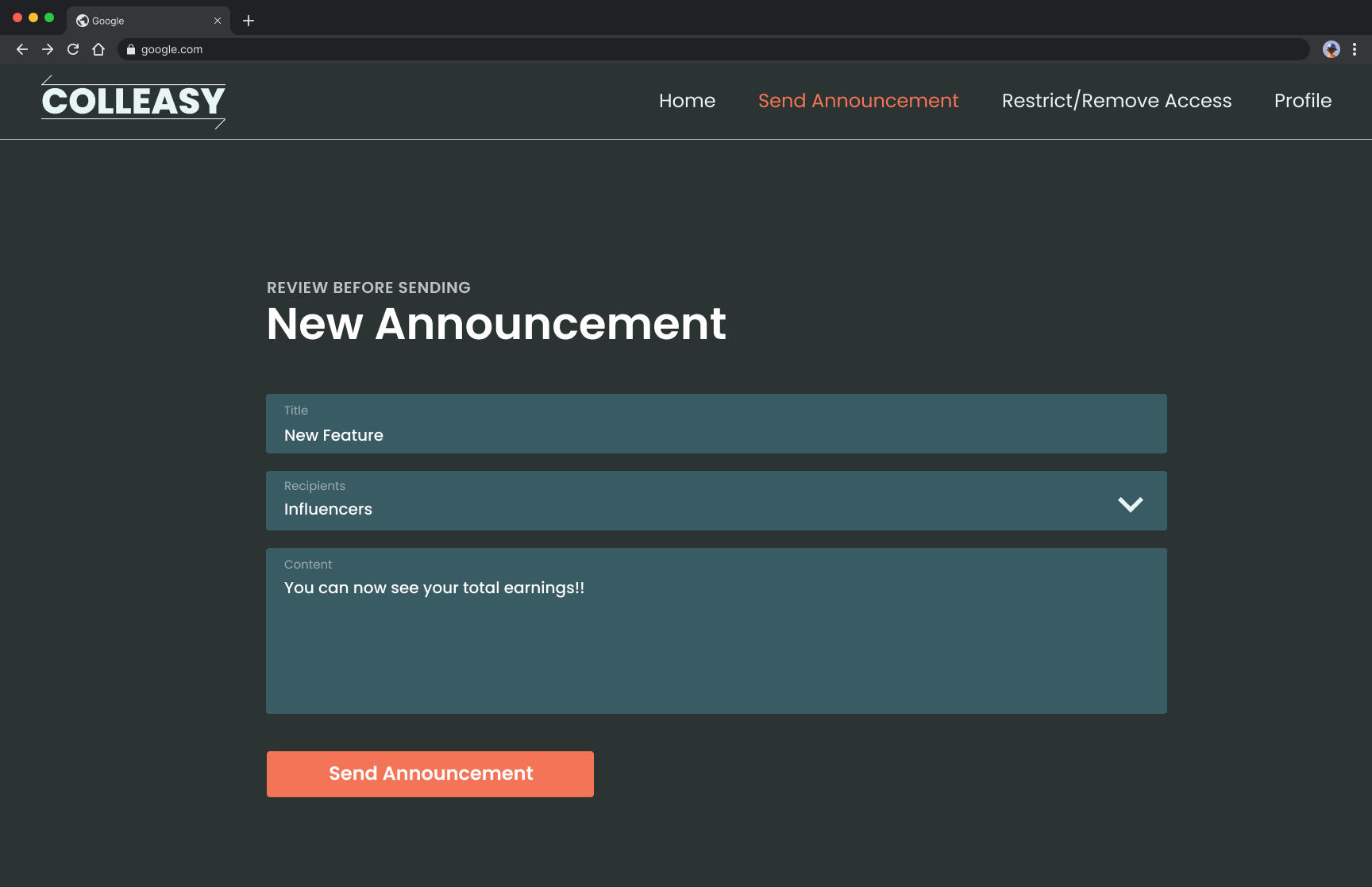
Alternatively, when the user clicks on login, they are navigated to the login screen where the user can enter his email and password to log in to their CollEasy account.



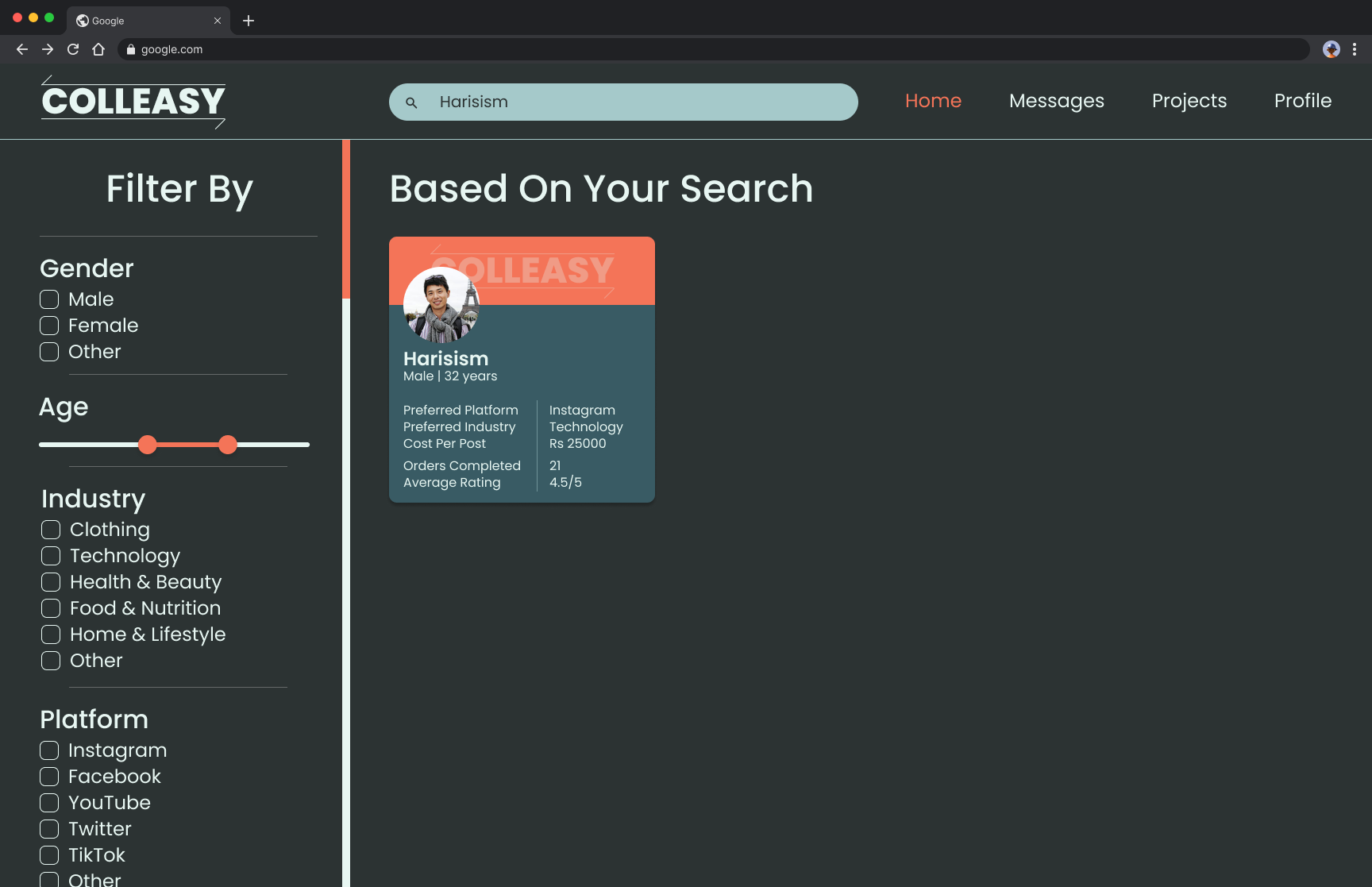
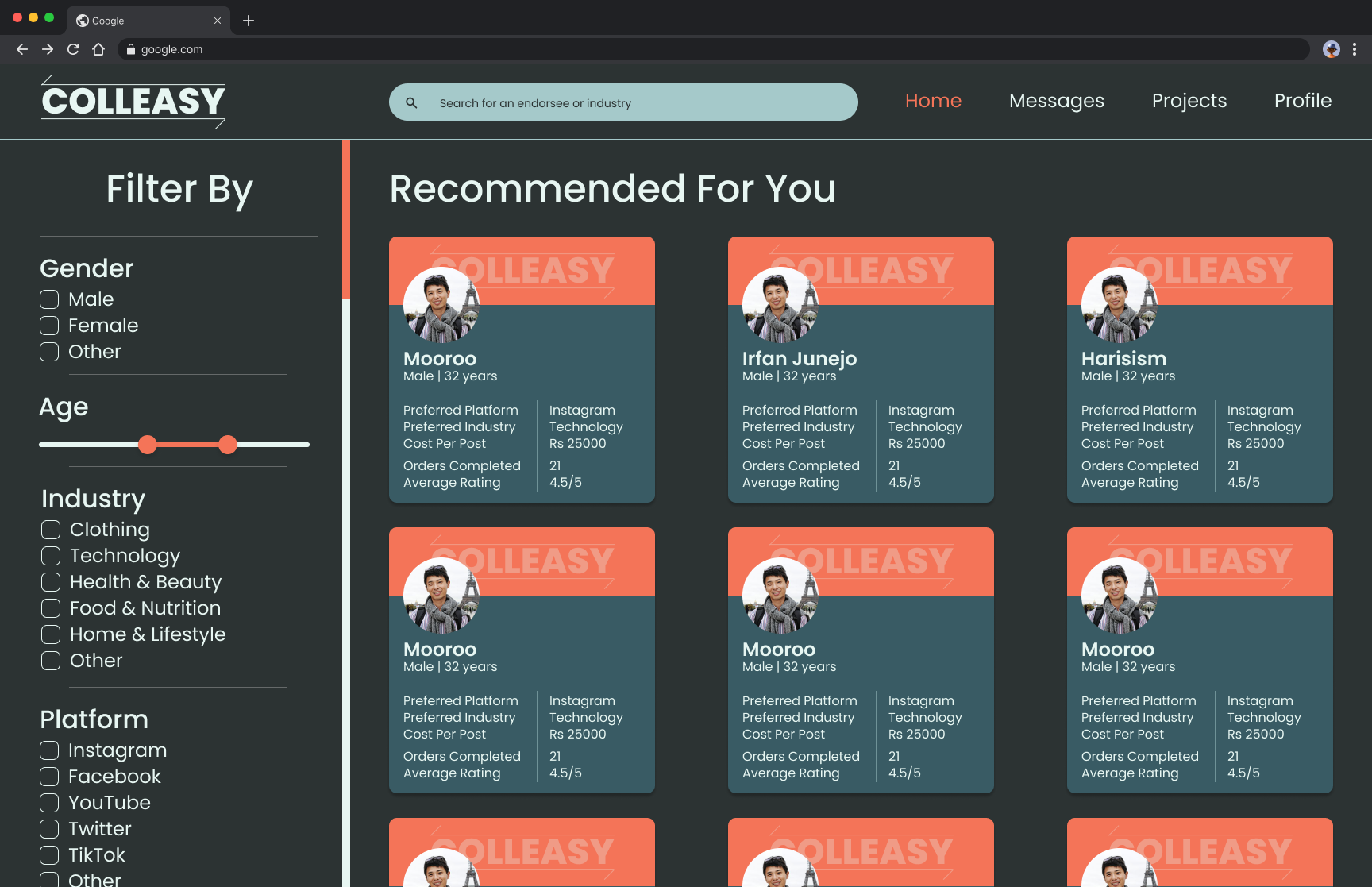
If the user has forgotten their password, they can click on Forgot Password. They will be then asked to enter their email address. Once they submit their email, they will be emailed a link to reset their password which will navigate them to a screen where they can enter and renter their new password to confirm password change.



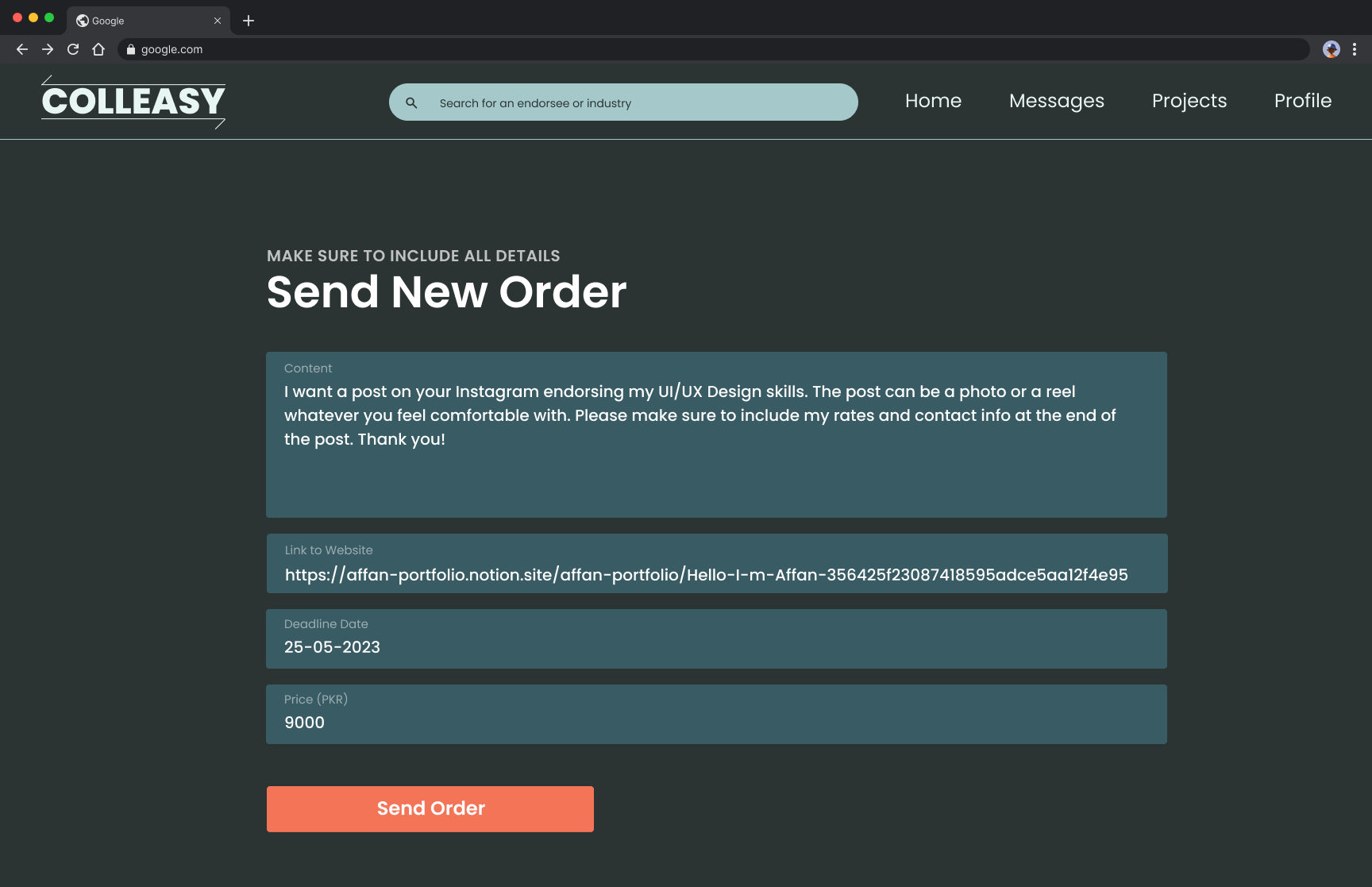
Similarly, the user can also change their password in the Profile tab in the navbar, Here, the user will be asked to enter their current password, their new password, and new password again. This way they can confirm that they want to change the password to what they want it to be.



The admin will be able to send an announcement to all users of the CollEasy website in Send Announcement tab in the navbar. There are 3 fields that the admin has to fill: 1) the title of the announcement, 2) the recipients of the announcement, and 3) the content of the announcement.



This is what the homepage looks like when an organization logs into their account. They are shown multiple endorsers recommended for them mainly based on their niche. The user can also search for an endorser through the search tab in the navbar.



The organization can click on an endorser and be directed to the Send Order screen where 4 fields are required to be filled in. The first one is the content or the message by the organization to the endorser regarding the opportunity. The second and third field is the link to the organization’s website and the deadline of the task respectively. The fourth and final field is the price or the amount that the organization is willing to pay to the endorser.

The endorser can then accept or reject the order. If the endorser accepts the order, once they have completed the task, they can go to their ongoing orders tab and mark it as completed. The organization now sees that the endorser has indeed completed the order and rates the work of the endorser. The average rating of the endorser is changed accordingly.

# Project Security

Brief introduction of this chapter in a paragraph

## Project Threats

Following are some of the threats to the app:

1. **Cross-Site Scripting (XSS) Attacks:** Using XSS attacks, malicious scripts can be injected into a web application, compromising user data and opening the door for more assaults.
2. **Attacks using SQL Injection:** Attackers can use SQL Injection to insert malicious SQL code into a web site, giving them access to the application's database without authorization.
3. **Cross-Site Request Forgery (CSRF) attacks:** CSRF can be used to deceive customers into making unauthorised purchases or changing their passwords on a website.
4. **DDoS attacks:** A web application can be rendered inaccessible to users by being overloaded with traffic thanks to Distributed Denial of Service (DDoS) attacks.
5. **Malware injection:** Malicious code can be injected into a web application using this technique, compromising user data and opening the door for more attacks.
6. **Poor authentication and authorization procedures:** Inadequate authentication and authorization procedures may allow unauthorized users to access the application and the data contained within.
7. **Inadequate security testing:** Inadequate security testing can leave the application's security with unknown flaws and vulnerabilities.
8. **Inadequate security measures:** The application may be exposed to assaults if there are insufficient security measures in place, such as weak passwords, a lack of encryption, and insufficient access controls.

## Potential Losses

Following are some of the potential losses to the app:

1. **Financial losses:** Security breaches can cause financial losses as a result of monies being stolen, transactions being made without authorization, and other types of financial fraud.
2. **Damage to reputation:** A security breach can harm the web application's and its owners' reputations, resulting in a decline in support from users, stakeholders, and the general public.
3. **Legal repercussions:** If it is discovered that the online application is in violation of privacy laws or other regulatory requirements, a security breach may result in legal repercussions, including fines.
4. **Downtime:** Downtime brought on by a security breach or other unanticipated circumstances can cost money, productivity, and client loyalty.
5. **Data loss:** Security lapses may cause the loss of delicate data, including financial information, client information, and intellectual property.
6. **System damage:** Security breaches may cause the servers, databases, and other elements of the online application's system to malfunction.
7. **Disruptions to operations:** Security breaches may result in lag times, service interruptions, and system outages.
8. **Business interruption:** Security breaches may seriously interfere with the web application's ability to do business, which could result in the loss of money, clients, and market share.

## Security Controls

1. **Secure coding procedures:** To lower the possibility of vulnerabilities in the application's code, developers should adhere to secure coding procedures. To stop SQL injection attacks, this entails employing parameterized queries, output encoding, and input validation.
2. **Authentication and authorization:** Strong authentication and authorization techniques must be implemented by the application to guarantee that only authorised users can access the application and its resources.
3. **Encryption:** To prevent unauthorised access, sensitive data, including passwords and financial information, should be encrypted when being stored and transmitted across a network.
4. **Access control:** Depending on the user's role and rights, access to the programme and its resources should be limited. This includes implementing least privilege access (LPA) and role-based access control (RBAC).
5. **Regular security audits:** To find and fix security flaws and holes in the application's security controls, regular security audits should be carried out.
6. **Plan for responding to security incidents and breaches:** The application should have a plan in place for responding to security breaches and other occurrences.
7. **Security of third-party components:** Any third-party components that are utilised in the programme should be constantly examined and updated to fix any security flaws.
8. **Secure application deployment and configuration:** Servers and other components should be configured securely with frequent security updates and patches, and the application should be deployed safely with the necessary security controls in place.
9. **Monitoring and logging:** To identify and quickly address security incidents, the application should be continuously monitored and logged.
10. **Validation of input:** Validation of input should be used to make sure that the application only accepts legitimate and expected data. This can aid in preventing vulnerabilities like command injection and cross-site scripting (XSS).
11. **Secure communication:** Data in transit between the application and the user's browser should be encrypted using secure communication protocols like HTTPS.
12. **Two-factor authentication:** To add a layer of protection beyond a password, two-factor authentication (2FA) should be used.

## Static and Dynamic Security Scanning Tools

**Static scanning tools:**

* A popular open-source static code analysis programme called SonarQube can be used to examine the source code of MERN stack applications. It offers a thorough set of guidelines and plugins to help find and report any bugs, security holes, and other problems with the code. SonarQube is a good tool for analysing MERN stack apps because it supports a wide range of programming languages, including JavaScript, React, and Node.js.
* A popular open-source static code analysis tool for JavaScript is ESLint. It may be used to enforce coding standards and find potential security holes in the code because it is very adaptable. ESLint is appropriate for examining MERN stack apps because it has numerous React and Node.js plugins available.
* The OWASP Code Review Guide is a thorough manual for examining the source code of web applications for security flaws. It makes it simple for developers to find and patch potential vulnerabilities in their code by providing a list of typical security issues and the related code examples.

**Dynamic scanning tools:**

* A well-known open-source dynamic scanning tool for web applications is OWASP ZAP. It can be used to uncover potential vulnerabilities and simulate real-world attacks to test the security of MERN stack applications. ZAP is an appropriate tool for evaluating the security of MERN stack applications since it offers a number of functions, including active scanning, passive scanning, and fuzz testing.
* Burp Suite: Burp Suite is another well-liked dynamic web application scanning tool. It is a great tool for evaluating the security of MERN stack applications since it offers a number of capabilities, including proxying, scanning, and testing. Burp Suite has a lot of customization options and may be used to mimic many types of attacks.
* A commercial dynamic scanning tool for web applications is called Acunetix. It is a useful tool for evaluating the security of MERN stack applications since it offers a variety of functionalities, including vulnerability screening, authentication testing, and compliance testing. Acunetix offers comprehensive reports on potential vulnerabilities and their severity and is capable of detecting a broad variety of vulnerabilities, including XSS, SQL injection, and directory traversal.

# Risk Management

## Potential Risks and 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. |

# Testing and Evaluation.

We have used an **End-to-End testing strategy** for our app. End-to-end testing (E2E testing) is a sort of software testing in which the complete system or application is evaluated from beginning to end. E2E testing ensures that the software system or application performs as expected and meets the business requirements.

E2E testing covers the entire programme, including the user interface, business logic, and database interactions. E2E tests replicate user interactions with the application and ensure that the system operates as expected.

E2E testing is an important aspect of the software testing process because it ensures that the application performs as intended in real-world circumstances and aids in identifying and preventing any errors before users become aware of them.

**Sample Test Cases:**

Here is the doc with sample test cases: [click here to view](https://pern-my.sharepoint.com/:x:/g/personal/23100268_lums_edu_pk/EZ0XuLUqvg5Mpa-ZYf1rvBUBmhLPM-L6BE49j4zrNCXnpg?e=K12cea&nav=MTVfezAwMDAwMDAwLTAwMDEtMDAwMC0wNDAwLTAwMDAwMDAwMDAwMH0)

We have used **Selenium for automated testing.**

Selenium is a free and open-source automated testing tool for web applications. It enables you to test your web application in a variety of scenarios by automating web browsers across several platforms and programming languages.

Selenium allows you to automate the interaction of a web browser with web pages by writing scripts in computer languages such as Java, Python, C#, and Ruby. To test the functionality and performance of a web application, it can imitate user actions such as clicking buttons, filling out forms, and browsing between pages.

# Deployment Guidelines

List down the steps for deployment of your system. Start from where the code (link of the github repository) should be picked and then mention all the steps for deployment in a production environment. Also mention the online link where your application is hosted along with access information (user/password etc.).

Code Repo Link : [Mbilalshahid10/P05-InstaShop (github.com)](https://github.com/Mbilalshahid10/P05-InstaShop)

Website deployed link : <https://colleeasy.herokuapp.com/>

So the step I followed were as follows:

1. Stored the repo to be deployed in a separate repo named : testing-prototype( naming it)
2. Login in to Heroku account and create a new app and name it accordingly. Then navigate to the deploy dashboard and connect our own repo with the account as shown in the image (Fig 1).
3. Moreover since changes can be updated we have enabled automatic deployments so the app is working after update and won't have to manually deploy again and again (Fig 2).
4. In Addition in our repo to build the app we make a package.json file and pass the path for the app to run build command as we did it automatically runs the server and launches the client as well.

"build": "cd server && npm install && cd ../client && npm install && npm run build"

1. Also since we are using Node.Js environment in the setting add a buildpack to run the scripts (Fig 3).
2. Once we have set up the whole process in order we manually deploy the process and it will give you the working deployed link as attached above. All possible data information is also attached in the README file in the given repo.

*Sample Login Data*

Admin login:

username : [bilal.shahid@gmail.com](mailto:bilal.shahid@gmail.com) password: admin123

Client login:

username: [rhs@gmail.com](mailto:rhs@gmail.com) password : client123

Influencer login:

username: [salman.masood@gmail.com](mailto:salman.masood@gmail.com) password : influencer123

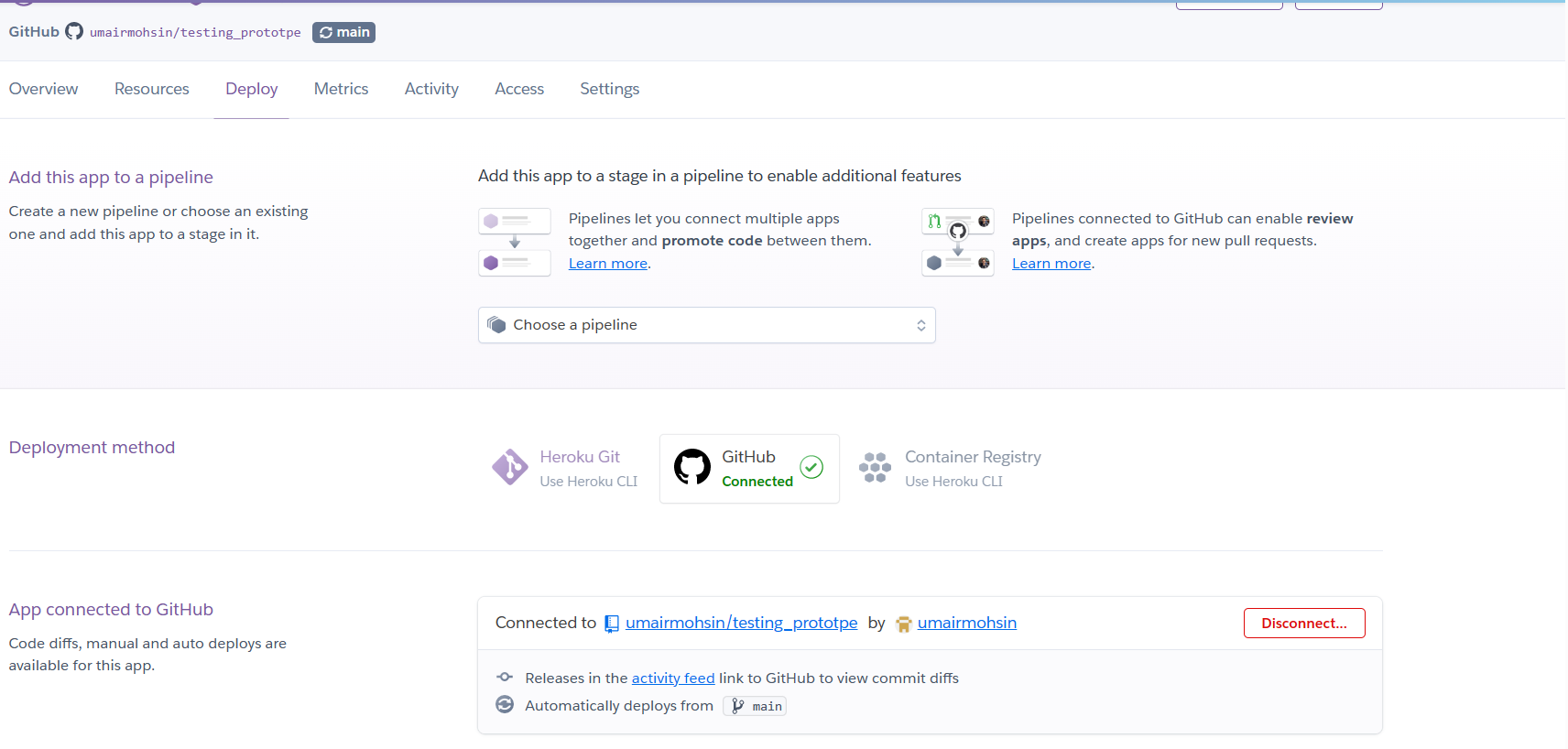
\

Fig 1 (Connect repo)

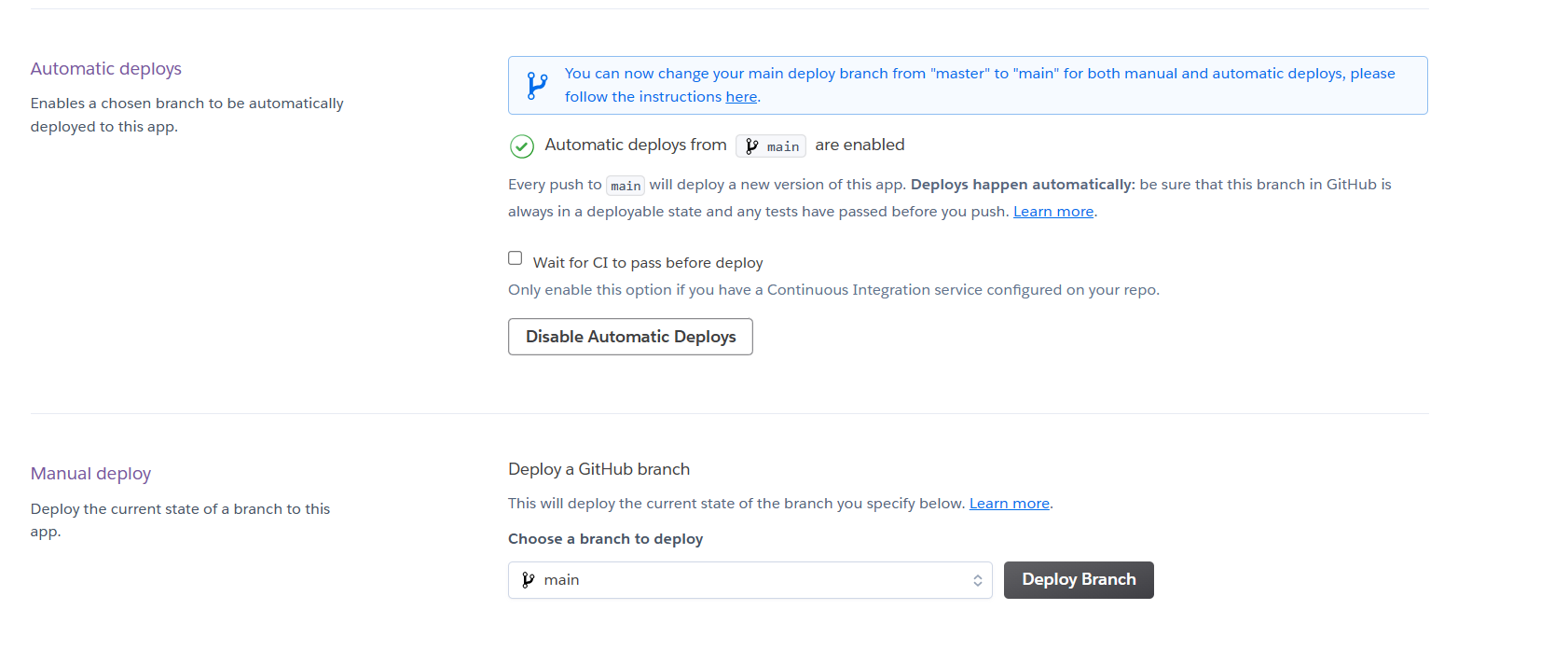


Fig 2 (Automatic deploy)

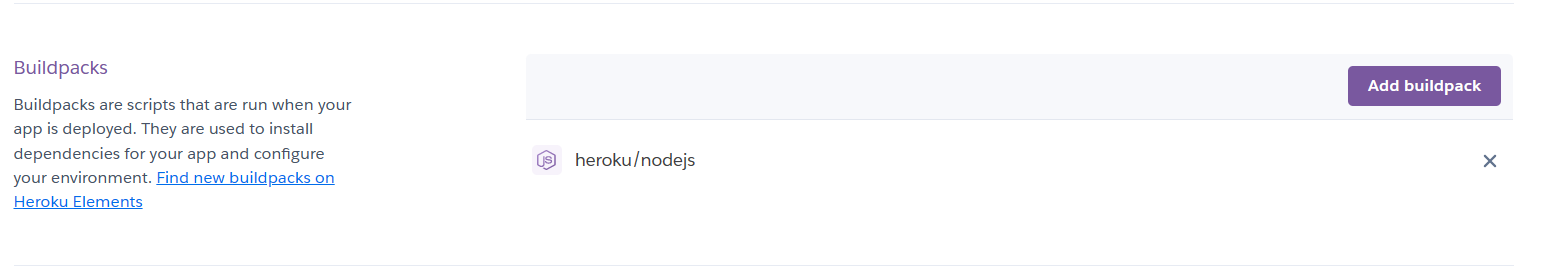


Fig 3 (Node.js Buildpack)

# Conclusion

## Summary

This sub-section will summarize how you have approached this project and what you have learned.

We have approached the project in the following way:

Plan the project; define the project scope, requirements, and objectives, then write a project plan outlining the timetable, budget, and deliverables. Design the application's architecture, including the frontend and backend technologies, as well as the database schema and API endpoints. Develop the application; implement the application's frontend and backend components, adhering to recommended practises for code organization, testing, and deployment. Thoroughly testing was performed to confirm that the application works as intended and meets the project requirements. Deploy the application; in a production environment, deploy the application using best practices for scalability, security, and performance.

Following are the key learnings of this project:

* **Understanding the architecture:** Understanding the architecture of the MERN stack and how the various components interact is critical for designing and deploying a successful application.
* **Best development practices:** Following best practices for code organization, testing, and deployment is critical for creating a high-quality application that satisfies project requirements and works well in production.
* **Importance of testing:** Thoroughly testing the application aids in identifying and preventing any flaws before users become aware of them.
* **Understanding deployment** factors such as scalability, security, and performance is critical for deploying the application to a production environment that fulfills project objectives.

## Challenges

Elaborate the issues and challenges, both technical and non-technical, faced during this project and how you have addressed them.

**Technical challenges:**

* There were compatibility concerns and faults when integrating different technologies, making it difficult to ensure a smooth integration.
* The performance of an application can suffer if it is not optimized for efficiency and speed.

**Non-technical challenges:**

* Creating a MERN stack application frequently involves several team members, which might cause communication issues.
* Developing a MERN stack application can have a lot of moving elements, making meeting project deadlines difficult.

**How to address them?**

By using the best coding practices, we overcame the technical challenges. And by having a clear communication among the group members, we overcame the non-technical challenges we faced during this project.

## Future

Our App is ready to be used as a marketplace for social media marketing. We have built a fully functional web-app to be used but we have placed the dummy data in it for now. Adding data to the app takes much effort in itself too. We have to take consent from the social media influencers and the clients for adding their data to our app. Now the next step can be collecting data for the different influencers and their clients and adding it to our app so it can work properly. We can also add some more features to our app like adding a live chat feature which can make the life of influencers and clients a lot easier. And as development is the never-ending process, we can add different updates to our app regularly with time.

# Review checklist

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

| **Chapter/Section Name** | **Reviewer Name(s)** |
| --- | --- |
| Introduction + Database design + deployment guidelines | Muhammad Bilal Shahid |
| System requirements + System user interface + summary | Muhammad Affan Ashraf |
| System architecture + Project security + challenges | Farva Talib |
| Requirements specifications + Risk management + Future work | Muhammad Umair Mohsin |
| Software development method + Testing and evaluation | Salman Masood |

# References