A  
PROJECT REPORT  
ON

Freelancing Site

By

SHINGALA SHUBHAM P. (CE-143) (19CEUOS159)UNAGAR KEVAL V. (CE-164) (19CEUBG010)

VAGHANI SMIT D. (CE-166) (19CEUEG022)

B.Tech CE Semester - VSubject : Advanced Technologies (CE - 515)

Guided by:

Prof. Prashant M. Jadav

Prof. Siddharth P. Shah



**Faculty of Technology  
Department of Computer Engineering  
Dharmsinh Desai University**

  
**Faculty of Technology  
Department of Computer Engineering  
Dharmsinh Desai University**

**CERTIFICATE**

This is to certify that the practical / term work carried out in subject

Of **Advanced Technologies** ( CE - 515 ) and recorded in this

journal is the Bonafide work of

SHINGALA SHUBHAM P. (CE 143) (19CEUOS159)

**UNAGAR KEVAL V. (CE 164) (19CEUBG010)**

**VAGHANI SMIT D. (CE 166) (19CEUEG022)**

Of B.Tech semester **V** in the branch of **Computer Engineering**

During the academic year **2021-22.**

Prof. Prashant M. Jadav Dr. C. K. Bhensdadia,

Prof. Siddharth P. Shah Head,

Dept. of Computer Engg., Dept. of Computer Engg.,

Faculty of Technology Faculty of Technology

Dharmsinh Desai University, Nadiad Dharmsinh Desai University, Nadiad

**Contents:**

1 Introduction...……………………….………………..…….….……………….. 4

2 Software Requirement Specifications.…………..……………..……….. 6

3 Design….…………………………….……………………..…….….………..….. 10

3.1 UseCase Diagram………….……………………….……..…………. 10

3.2 Data flow Diagram……………………………………………………. 11

3.3 Sequence Diagram…………………………………………. 12  
3.4 Data Dictionary……………………………………………… 14

4 Implementation Details ……………………………….…………..……….… 16

5 Screen-shots ……………………………………………………….…………….. 22

6 Conclusion …………………………………………….………………...…….…. 30

8 Limitation and Future Extension ………………………………………….. 31

9 Bibliography ……………………………………………..………………………. 31

1. **Introduction**

Freelancing Site is an online website that can be used by Freelancers

and Buyers, where buyer can upload the project ( task ) with required

details like description of project, duration days, budget, etc. Freelancer can

see different type of project on site and can bid on project relevant to their

category. After seeing all the bids on the project, buyer can hire one of the

freelancer for their work. After Freelancer submit the work, buyer will have

to complete the project from their side, then freelancer will get money.To

use any of the functionalities user must have to register / login on site.

The first module is registration as buyer or freelancer. in this module

buyer or freelancer can register their self on website by providing all the

information. The Second module is Profile. In this module Buyer and

Freelancer can see their profile and update it whenever required.

Modules Specific to buyer :- 1) Post Project – In this module buyer upload

project with required details. 2) My Project – In this module buyer can see

their uploaded project & manage it like seeing bids by freelancers, delete

project, hire any freelancer, status etc.

Modules Specific to freelancer :- 1) Find Jobs – In this module freelancer

can see all the uploaded project by buyers which are relevant to their

category and bid on them. 2) My Project – In this module freelancer can see

all their active or completed projects.

* **Tools/Technologies Used**
* **Technologies:**
* HTML
* CSS
* Bootstrap
* Javascript
* Typescript
* Angular
* Node Js
* Express Js
* MongoDB
* **Tools**
* Git
* Visual Studio Code

* **Platform**
* Local development server

1. **Software Requirement Specifications**

**Users of the System :-**

* Buyer
* Freelancer

**Functional requirements :-**

**1. Login / Registration**

**R.1.1: Create New Account  
Description :-** If user does not have an account on website then he/she has to create an Account.  
**Input:-** Enter details like name, email id, password, type of account (i.e. buyer or freelancer) etc... to create an account.  
**Output:-** Redirected to Login

**R.1.2: Login  
Description :-** User can access this website by login using email id and password.  
**Input:-** Enter email id and password.  
**Output:-** If email id and password are correct then all functionalities will be display, otherwise message will be display as “Invalid email id or password”.

**To Use Any Of the below module & functionalities user must require login.**

**2. Profile**

**R.2.1: See Profile  
Description :-** User can see their Profile Details like name, description, category, education, certification, profile image and update it.

**R.2.1.1: Upload Profile Picture  
Input:-** User have to select profile picture from their computer.  
**Output:-** Uploaded image will be display as profile picture.

**R.2.1.2: Modify Personal Details  
Input:-** User can select and modify details.  
**Output:-** Details will be modified.

* **Modules Specific To Buyers**

**1. Buyer’s Project**

**R.1.1: Post Project  
Description:-** Buyer can post a project to hire freelancer.   
**Input:-** Buyer have to enter all the required details like name, description, category, budget & duration of project.  
**Output:-** Project will be uploaded.

**R.1.2: Delete Project  
Description :-** Buyer can select any of their uploaded project and delete it.  
**Input :-** Select Project & Press on delete button.  
**Output :-** Project will be deleted.

**2. My Project**

**R.2.1: See all project  
Description :-** Buyer can see their all uploaded projects.  
**Input :-** Go to my project section.  
**Output :-** All Uploaded project will be shown.

**R.2.2: See all Bids for particular project.  
Description :-** Buyer can selectone of their uploaded projects and see all the bids done by freelancers.  
**Input :-** Select project -> go to proposals.  
**Output :-** All bids will be shown.

**R.2.3: Hire Freelancer  
Description :-** After Seeing all the bids, Buyer can select one of the best bids and hire that freelancer.  
**Input :-** Select Project -> go to proposals -> hire freelancer  
**Output :-** Freelancer will be hired & get hired status from buyer. Remaining bids / proposals will be deleted automatically and no other freelancer can place bid on that project.

* **Modules Specific To Freelancers.**

**1. Find Jobs**

**R.1.1: Find Project  
Description :-** Freelancer can see all the uploaded projects by buyers relevant to their work category.  
**Input:-** Go to Find Jobs.  
**Output:-** All pending projects relevant to their work category will be displayed.

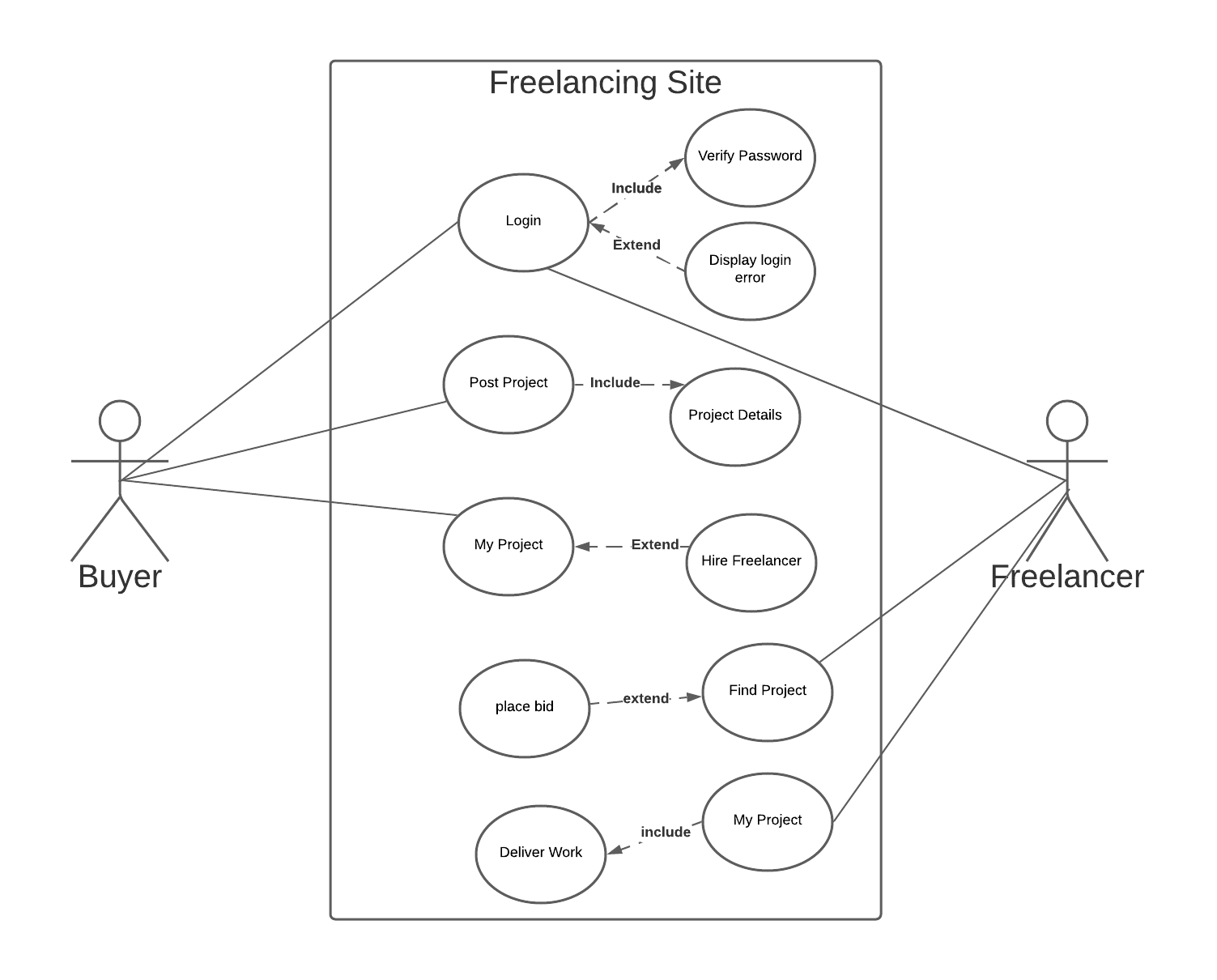
**R.1.2: Place Bid  
Description :-** Freelancer can select one of the project and place a bid for that project with required details.  
**Input :-** Enter bid details like bid amount, duration days, proposals.  
**Output :-** Bid will be placed for that project.

**2. My Project**

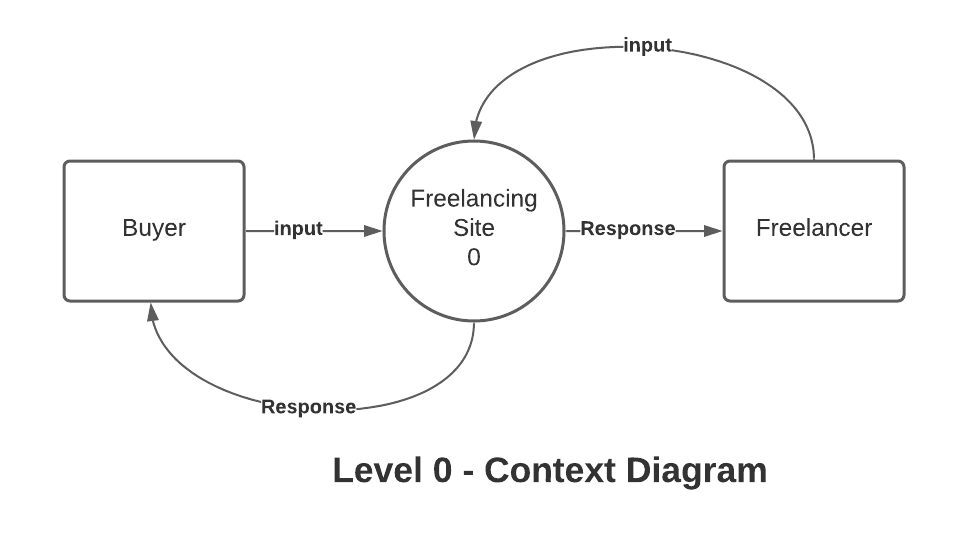
**R.2.1: My Project  
Description:-** Freelancer can see all their completed and active projects.  
**Input:-** Go to My Projects.  
**Output:-** List of Projects Will be displayed.

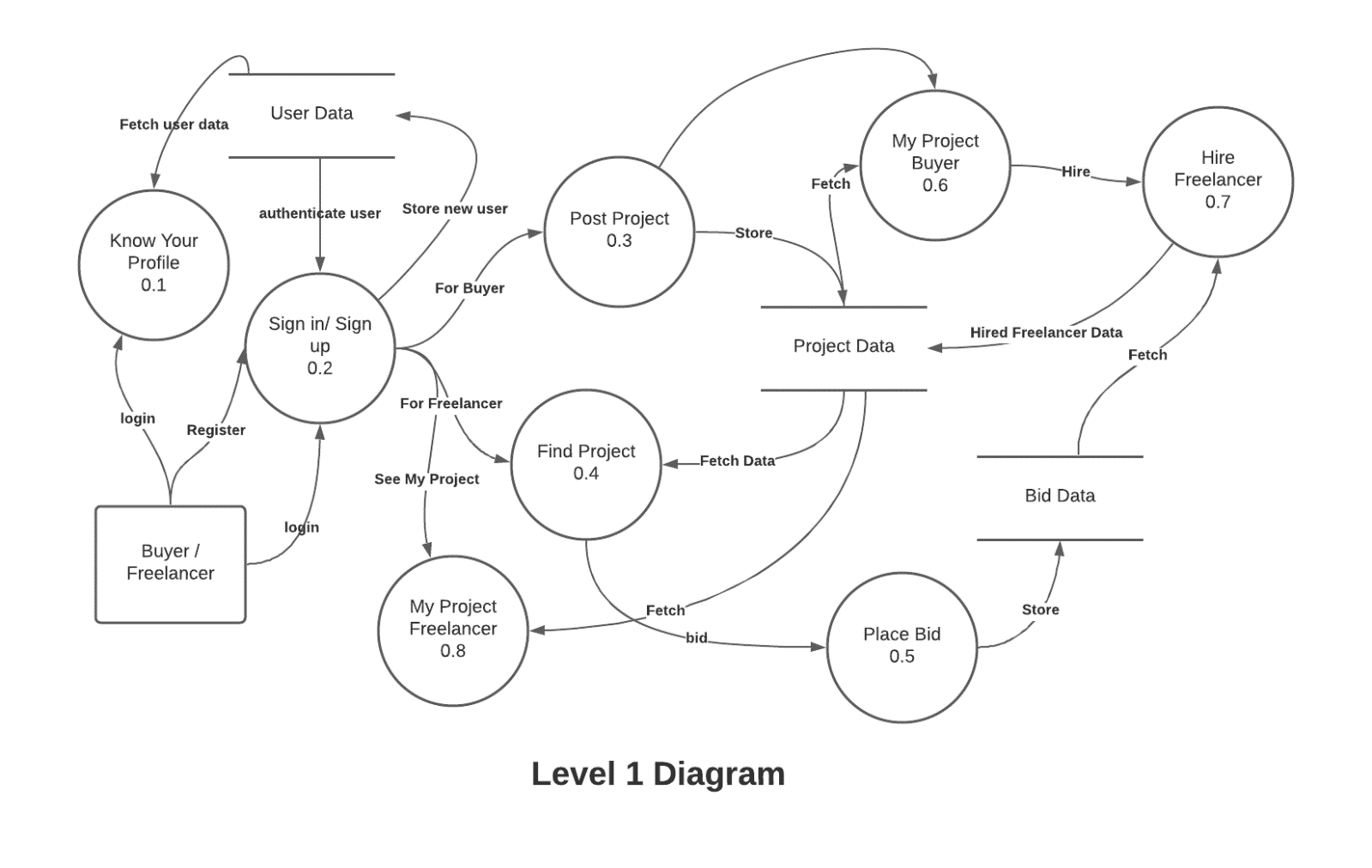
**R.2.1: Deliver Work  
Description:-** Freelancer can go to any active project and attach their work.  
**Input:-** Attach Work Files.  
**Output:-** File will be uploaded.

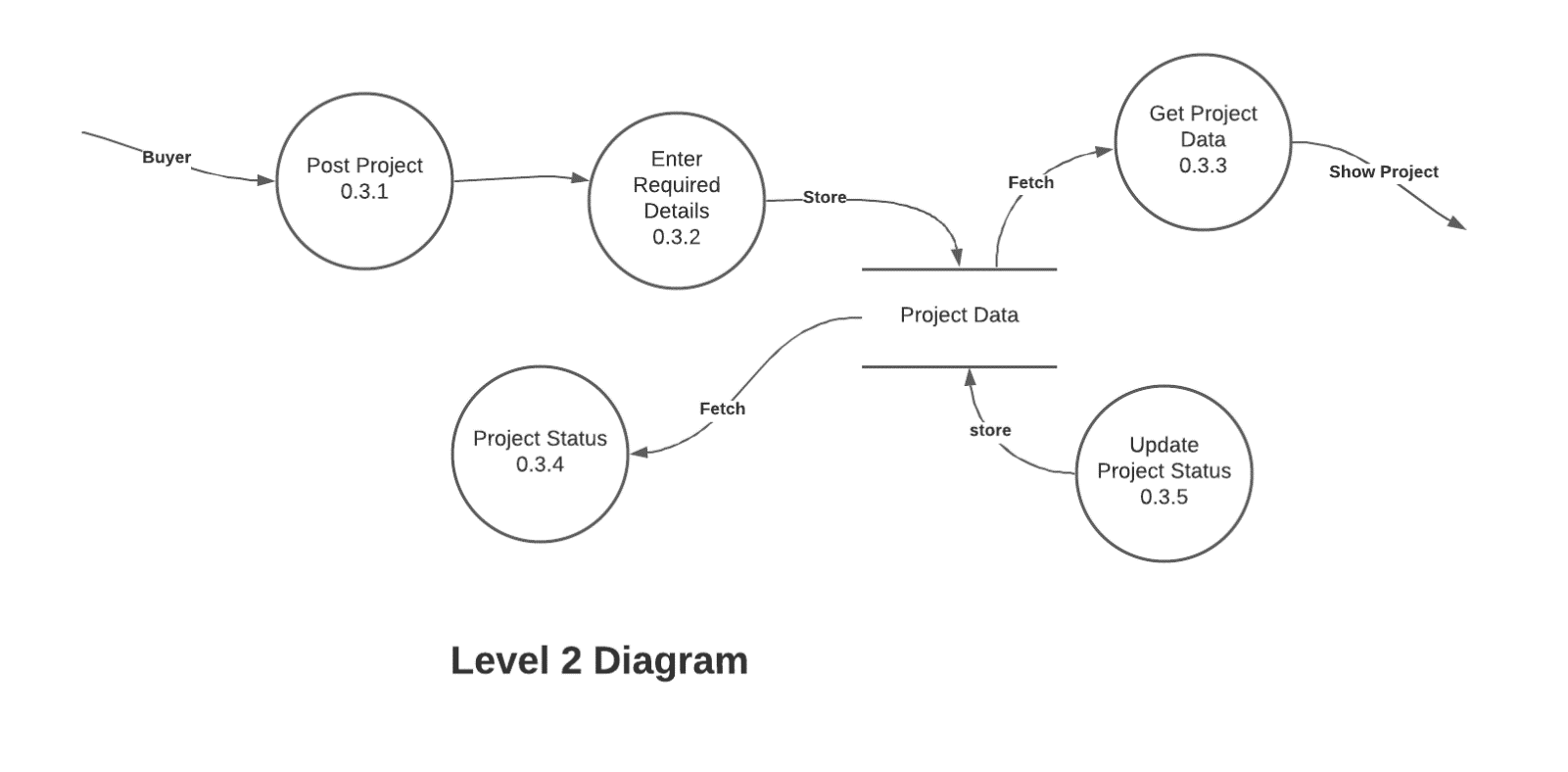
1. **Design** 
   1. **Use Case Diagram**



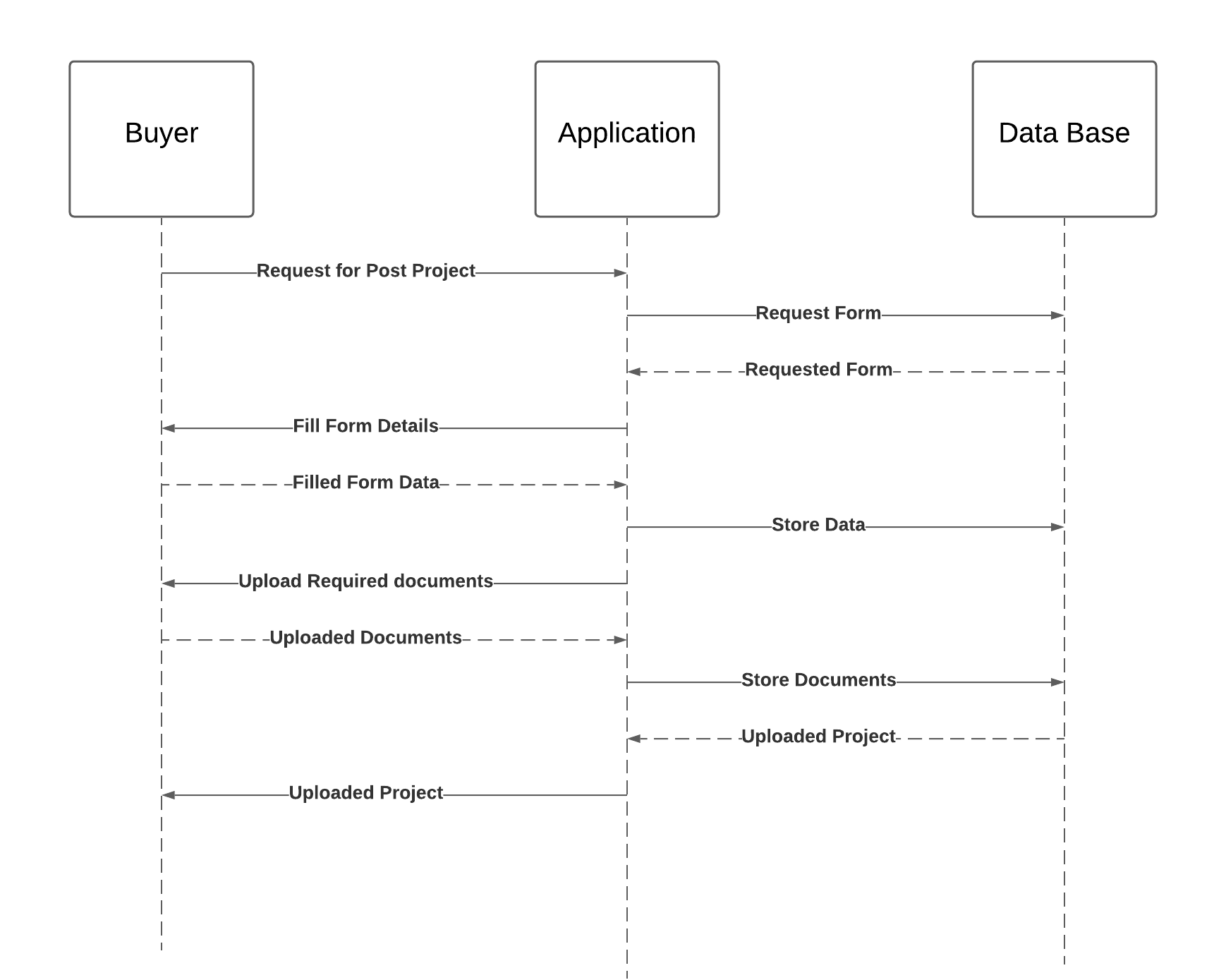
* 1. **Data Flow Diagram**



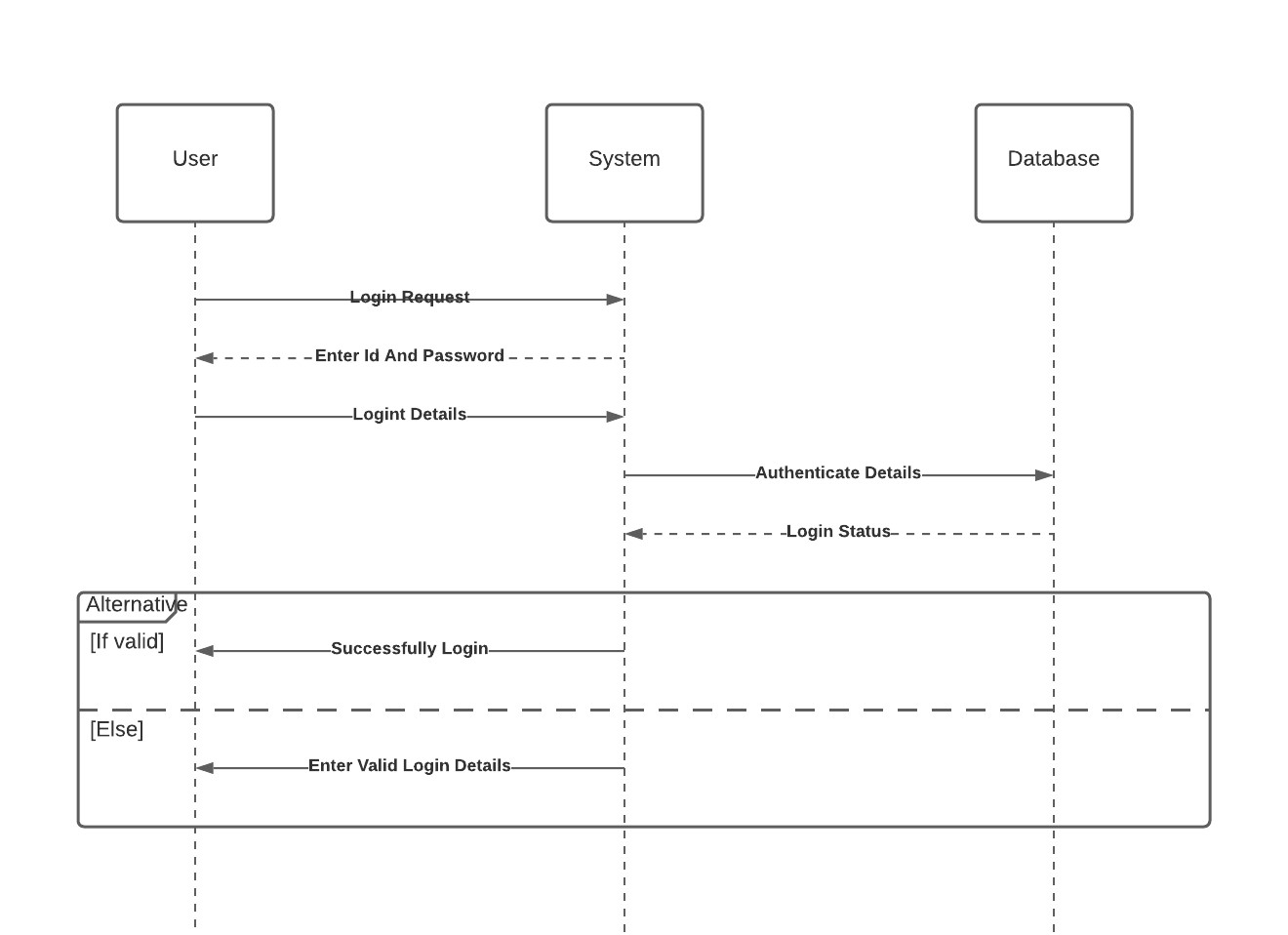




* 1. **Sequence Diagram (**Post A Project**)**



Login



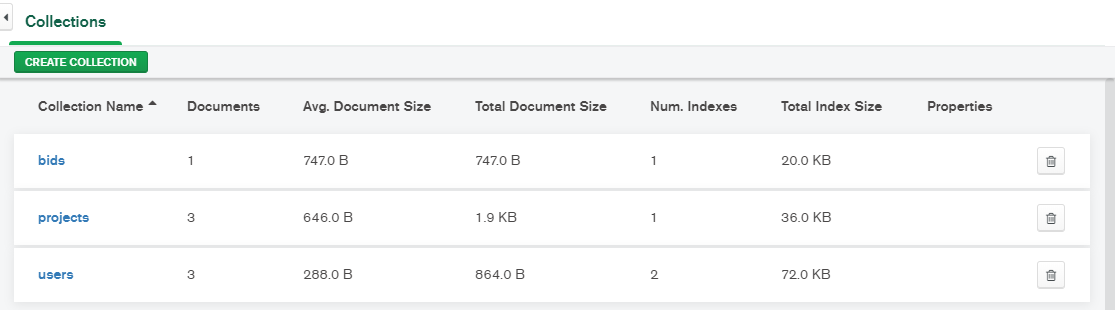
* 1. **Data Dictionary**

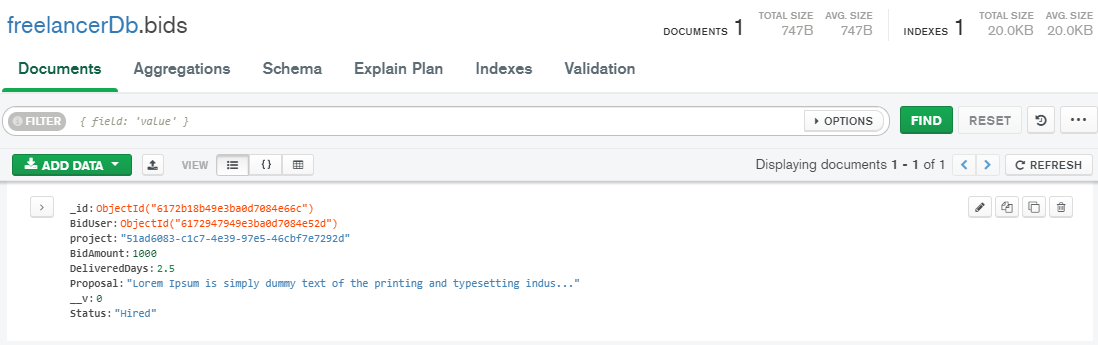
In our project we have used the NoSQL database so we cannot

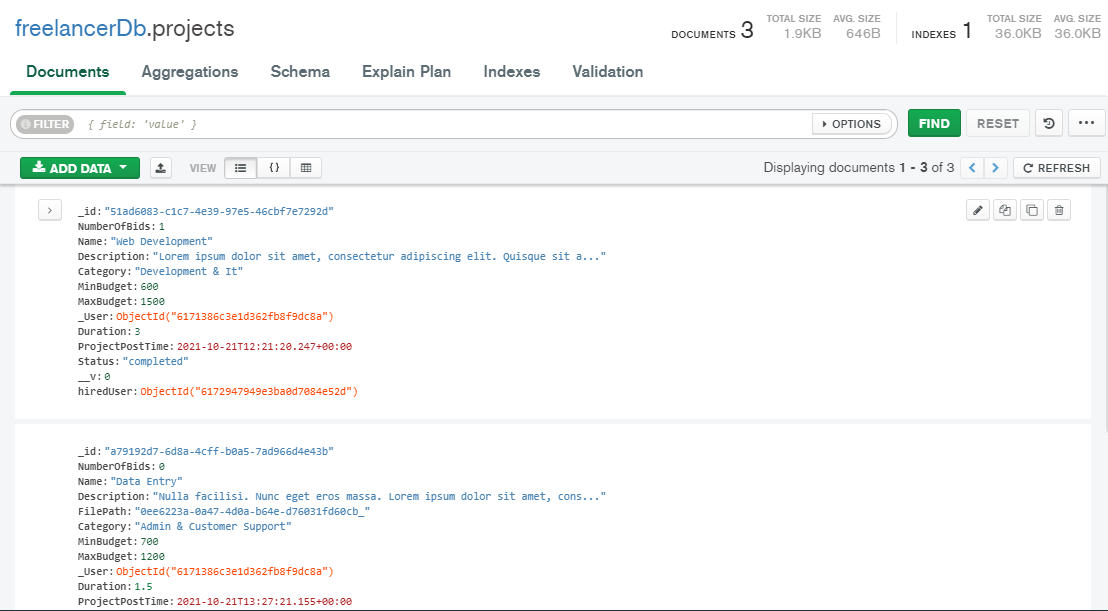
create a data dictionary as well as entity relationship diagram but we are

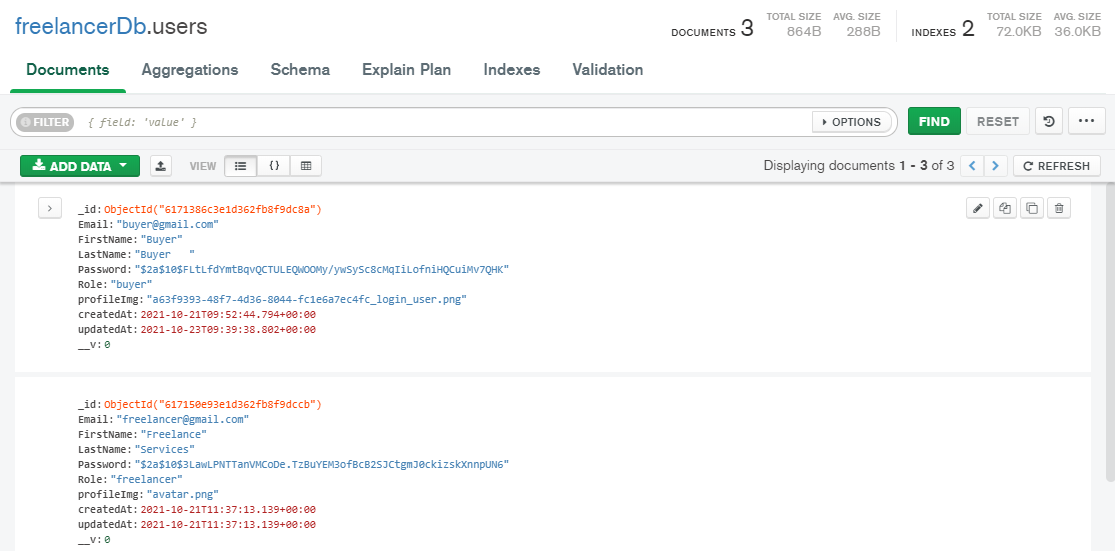
providing here some pictures of our database. We have used the MongoDb

Community Server.









1. **Implementation Details**

* Post a Project :-

onSubmit() {

    this.PostProjectForm.markAllAsTouched();

    if (this.projectName?.valid && this.description?.valid && this.maxBudget?.valid

      && this.minBudget?.valid && this.category?.valid && this.projectFile?.valid && this.duration?.valid) {

        this.authService.loggedUser().subscribe(

          (*res*:any)=>{

            if(res.status=='ok'){

              let date=new Date();

              var project: IProject = {

                \_id:uuidv4(),

                Name: this.projectName?.value,

                Description: this.description?.value,

                FilePath: this.uniqueFileName,

                Category: this.category?.value,

                MinBudget: this.minBudget?.value,

                MaxBudget: this.maxBudget?.value,

                \_User:res.data.\_id,

                Duration:this.duration?.value,

                ProjectPostTime:date,

                Status:"pending",

                NumberOfBids:0

              }

        this.projectService.addProject(project).subscribe(

                (*res*) => {

                  this.router.navigateByUrl('/project');

                },

                (*err*) => {

                  console.log(err);

                }

              )

            }

          }

        )

    }

  }

* File upload in post a project form :-

onFileSelected(*event*: any) {

    const file: File = *event*.target.files[0];

    if (file) {

      this.uniqueFileName=uuidv4()+"\_"+file.name;

      const formData = new FormData();

      formData.append("file", file,this.uniqueFileName);

      const upload$ = this.http.post(`${this.baseUri}/projectUpload`, formData, {

        reportProgress: true,

        observe: 'events'

      }).pipe(finalize(()=>this.reset()))

      this.uploadSub = upload$.subscribe(*event* =>{

        if (*event*.type == HttpEventType.UploadProgress) {

          if(*event*.total)

          this.uploadProgress = Math.round(100 \* (*event*.loaded /*event*.total));

        }else if (*event*.type === HttpEventType.Response) {

          this.fileName = file.name;

          this.reset();

        }

      },)

    }

  }

* Buyers uploaded project :-

getAllProject(){

    this.authService.loggedUser().subscribe(

      (*res*:any)=>{

        if(*res*.status=='ok' && *res*.data.Role=='buyer'){

          this.router.navigateByUrl('/')

        }

        if(*res*.status=='ok'){

          if(*res*.data.Category=='none'){

            this.projectService.getAllProject().subscribe(

              (*res*:any)=>{

                if(*res*.status=='ok'){

                  if(*res*.data.length==0){

                    this.error="Buyers haven't yet post project.";

                  }

                  else

                  this.projectList=*res*.data;

                }

              }

            )

          }

          else{

            let obj:any={

              category:*res*.data.Category

            }

            this.projectService.getProjectsOfCategory(obj).subscribe(

              (*res*:any)=>{

                if(*res*.status=='ok'){

                  if(*res*.data.length==0){

                    this.error="Buyers haven't yet post project In your Cateogry. If You want to Change Category then go to profile section.";

                  }

                  else

                  this.projectList=*res*.data;

                }

              }

            )

          }

        }

      }

    )

  }

* Place Bid by Freelancer :-

placeBid(){

    this.bidForm.markAllAsTouched();

    if(this.bidForm.valid){

      this.projectService.getOneProject(this.id).subscribe(

        (*res*:any)=>{

          if(*res*.status=='ok'){

            this.numberOfBid=*res*.data.NumberOfBids

          }

        }

      )

      let obj:IBid={

        BidUser:this.userId,

        project:this.id,

        BidAmount:this.bidAmount?.value,

        DeliveredDays:this.days?.value,

        Proposal:this.proposal?.value

      }

    this.bidService.placeBid(obj).subscribe(

      (*res*:any)=>{

        if(*res*.status=='ok'){

          let bid=this.numberOfBid+1;

          console.log(bid);

          let Obj={

            id:this.id,

            bid:bid

          }

          this.projectService.updateBid(Obj).subscribe(

            (*res*:any)=>{

              if(*res*.status=='ok'){

                this.router.navigate(['/findjobsProjects/',this.id,'Proposals']);

              }

            }

          )

        }

      }

    )

    }

  }

* Get all bids of project :-

getAllBids(){

    this.projectService.getOneProject(this.id).subscribe(

      (*res*:any)=>{

        if(*res*.status=='ok' && (*res*.data.Status=='Hired' || *res*.data.Status=='completed')){

          this.hiredProject=true;

          this.bidService.getBidOfHiredProject(this.id).subscribe(

            (*res*:any)=>{

              if(*res*.status=='ok'){

                this.bids=*res*.data;

              }

            }

          )

        }

        else{

          this.bidService.getBidOfProject(this.id).subscribe(

            (*res*:any)=>{

              if(*res*.status=='ok'){

                if(*res*.data.length==0){

                  this.error="No one Place Bid Yet!";

                }

                else

                this.bids=*res*.data;

              }

            }

          )

        }

      }

    )

  }

* Freelancer’s hired projects :-

getHiredProject(){

    this.authService.loggedUser().subscribe(

      (*res*:any)=>{

        if(*res*.status=='ok'){

          this.userId=*res*.data.\_id;

          this.projectService.getAllHiredProject(this.userId).subscribe(

            (*res*:any)=>{

              if(*res*.status=='ok'){

                {

                  if(*res*.data.length==0)

                  {

                    this.error="No Buyer has hired you yet!";

                  }

                  else

                this.projectList=*res*.data;

                }

              }

            }

          )

        }

      }

    )

  }

* Project file upload api :-

route.post('/projectUpload',(*req*, *res*,*next*)=> {

    let file = *req*['files'].file;

    let uploadPath=path.join(\_\_dirname,"uploads")+"/"+file.name;

    file.mv(uploadPath,(*err*)=>{

        if(*err*){

            return next(*err*);

        }

        else{

            return *res*.json({status:'ok',data:'file uploaded'});

        }

        })

  });

* Login user api :-

route.post('/login',(*req*,*res*)=>{

    user.findOne({Email:*req*.body.Email},(*err*,*data*)=>{

        if(*err*){

            console.log(*err*);

        }

        else{

            if(!*data*){

            return *res*.json({status:'error',error:'invalid email/password'})

            }

            comparePassword(*req*.body.Password,*data*.Password,(*flag*)=>{

                if(*flag*){

                    const token=jwt.sign({

                        id:*data*.\_id,

*// Email:data.Email,*

*// Role:data.Role*

                    },config.TOKEN\_SECRET)

                    return *res*.json({status:'ok',data:token})

                }

                else{

                    return *res*.json({status:'error',error:'invalid email/password'});

                }

            });

        }

    })

})

* Register user api :-

route.post("/register", (*req*, *res*, *next*) => {

    let Obj = *req*.body;

    if(!Obj.Email){

        return *res*.json({status:'error',error:'email required'})

    }

    const ps = bcrypt.hash(Obj.Password, 10, (*err*, *pass*) => {

        if (*err*) {

            return next(*err*);

        }

        user.create({

            Email: Obj.Email,

            FirstName: Obj.FirstName,

            LastName: Obj.LastName,

            Password: *pass*,

            Role: Obj.Role,

            profileImg:Obj.profileImg

        }, (*err*, *data*) => {

            if(*err*){

                if (*err*.code==11000) {

*// console.log(Obj.Email);*

                    return *res*.json({status:'error',error:'email already in use'});

                }

                else

                return next(*err*);

            }

            else

            return *res*.json({status:'ok'})

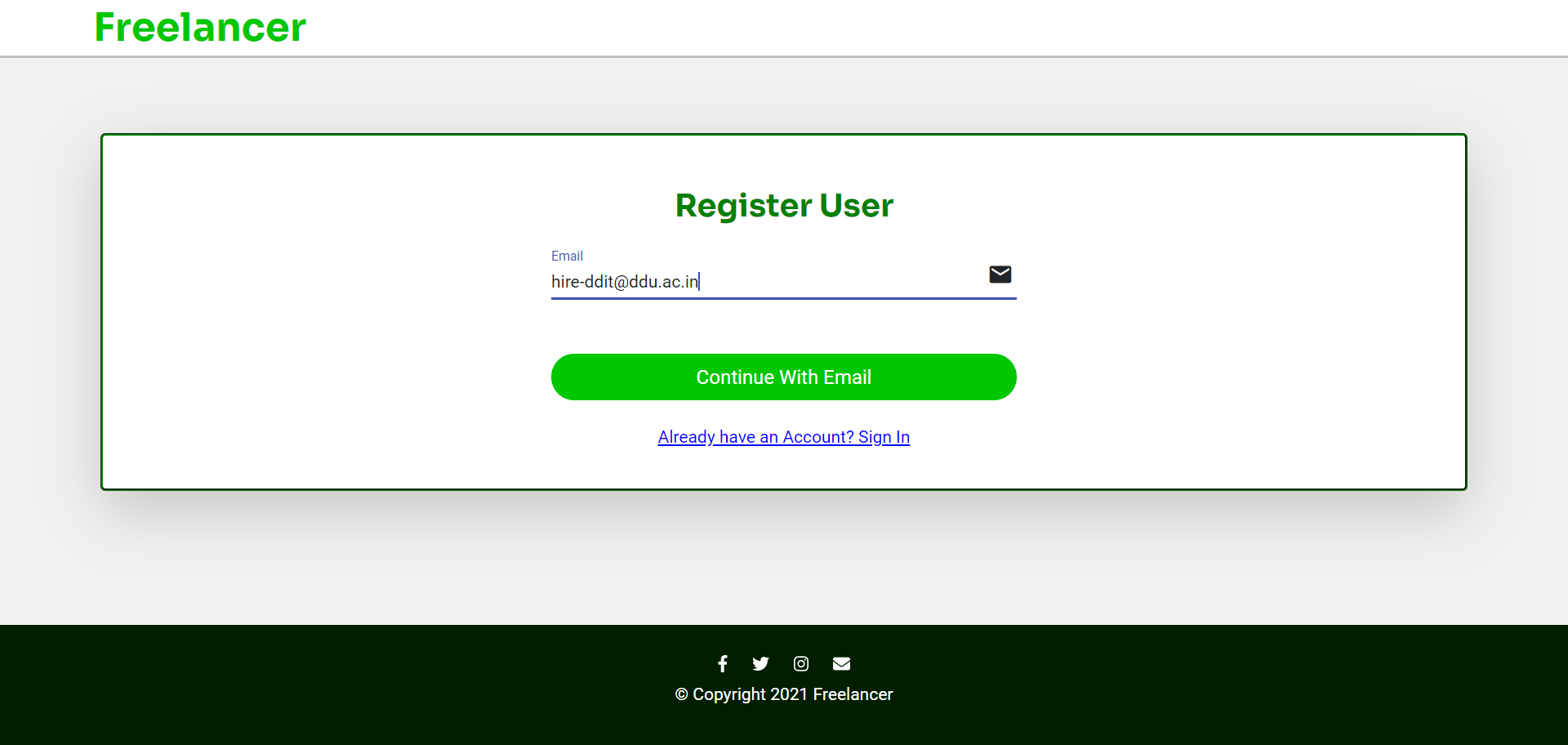
        })

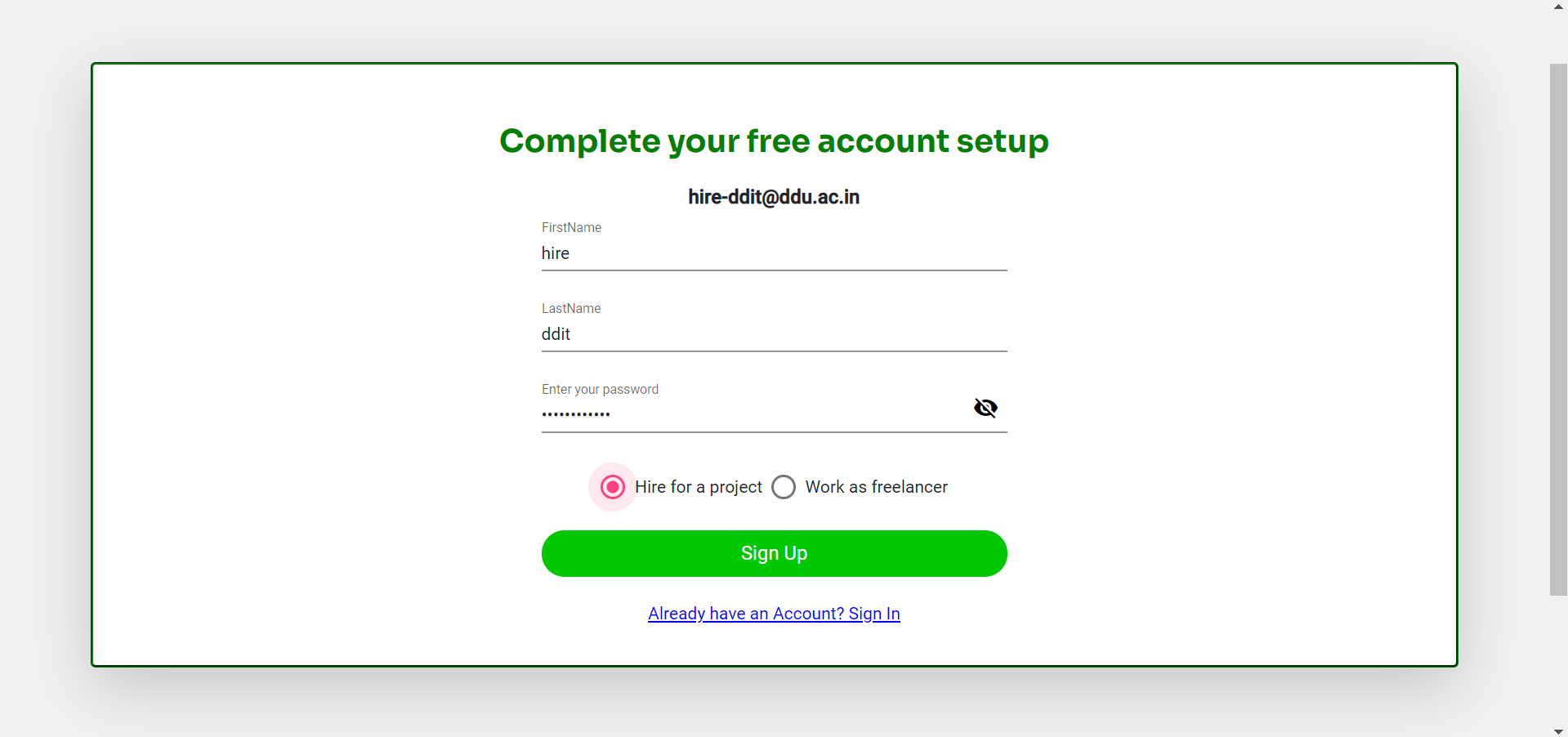
    });

})

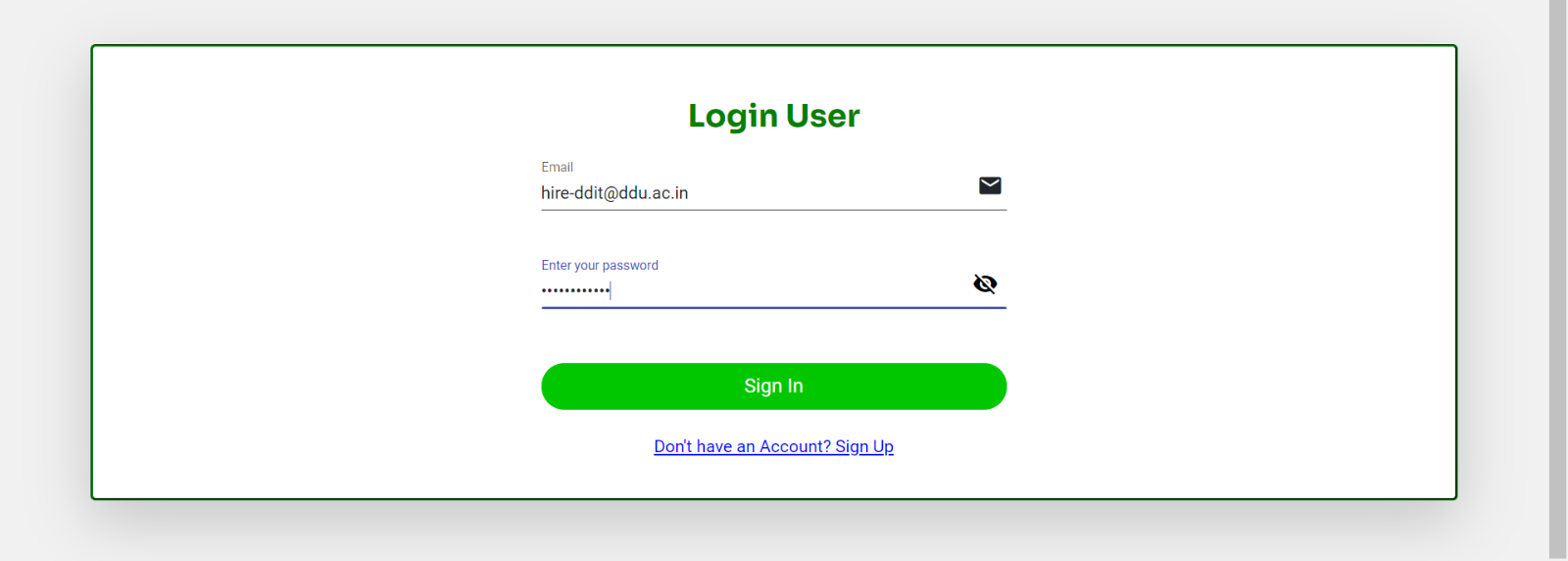
1. **Screenshots**

* Register:-

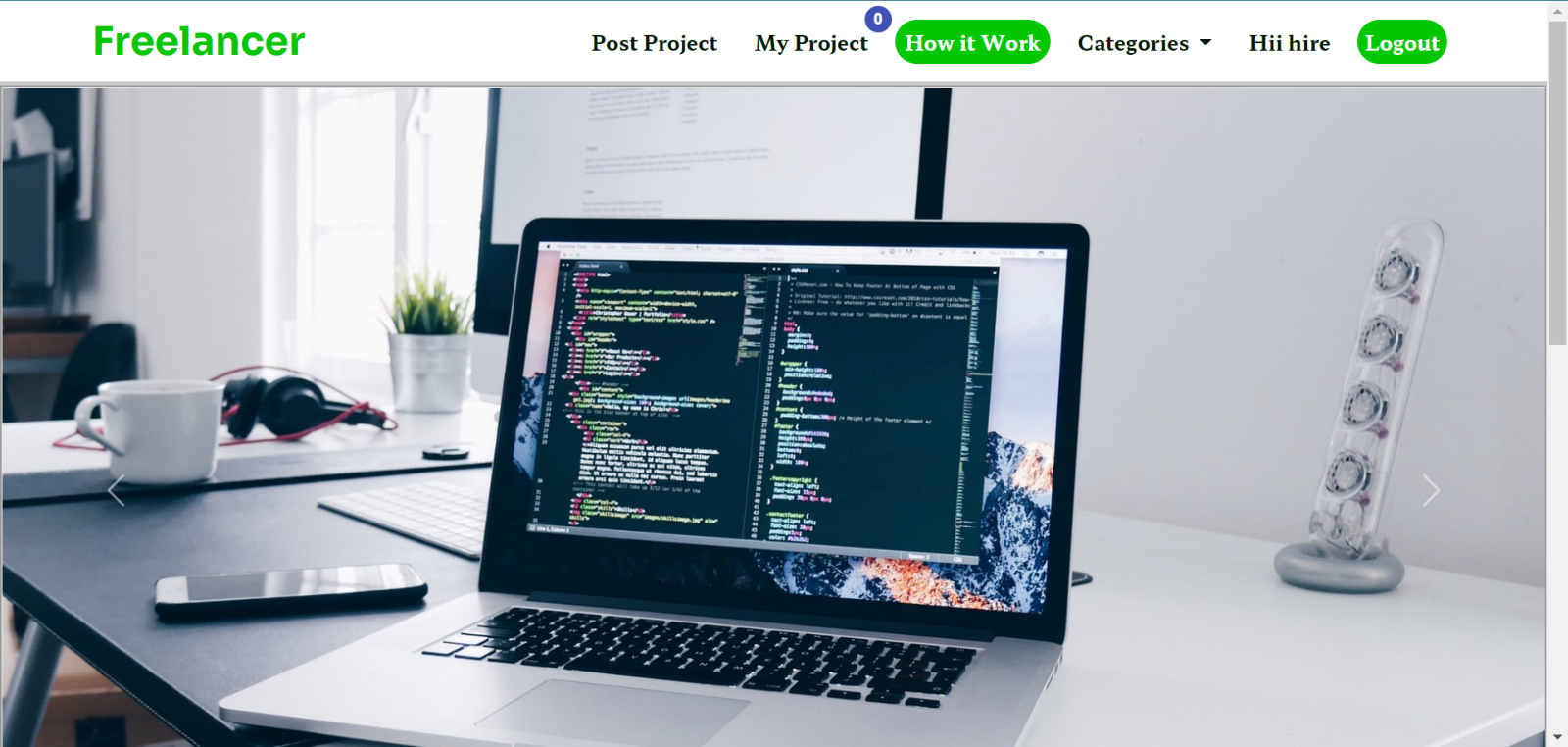




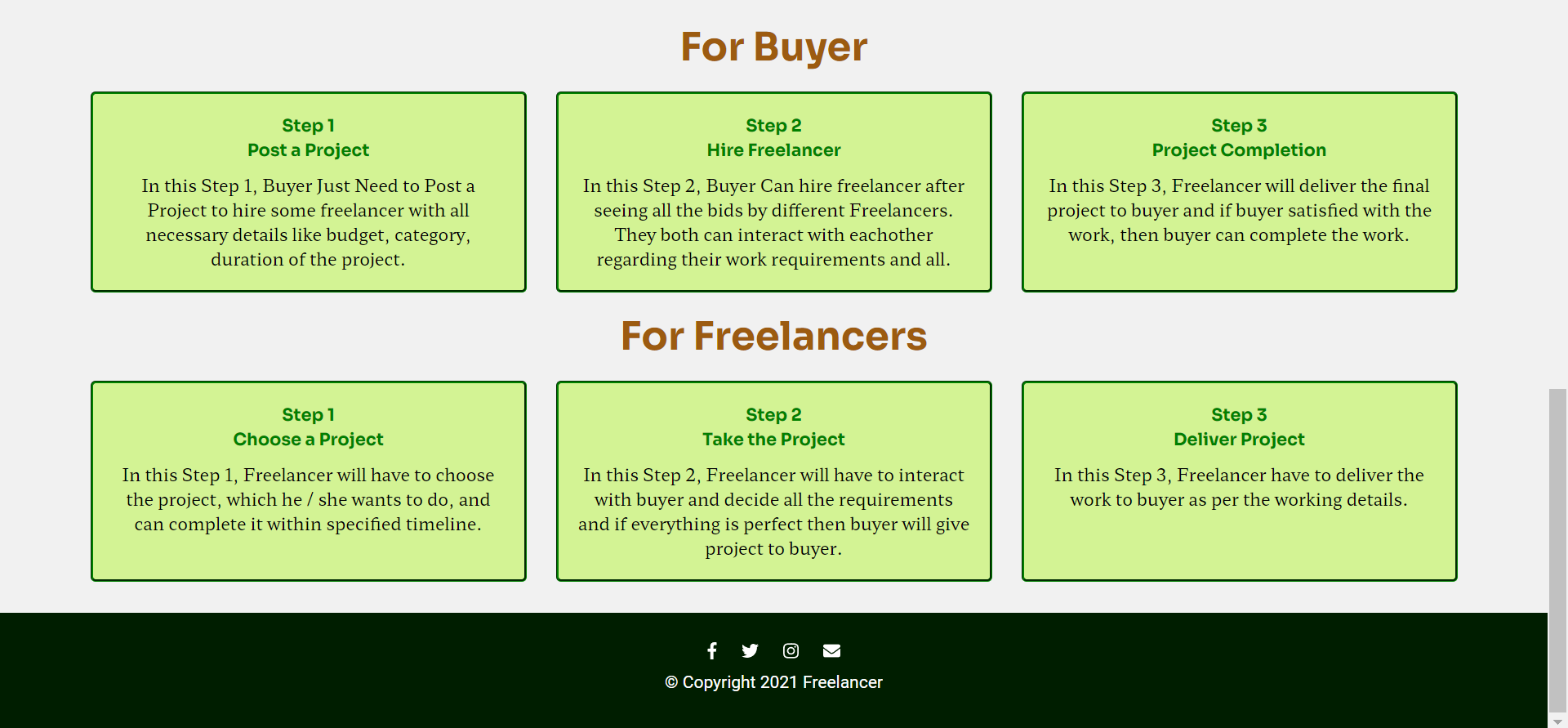
* Login:-



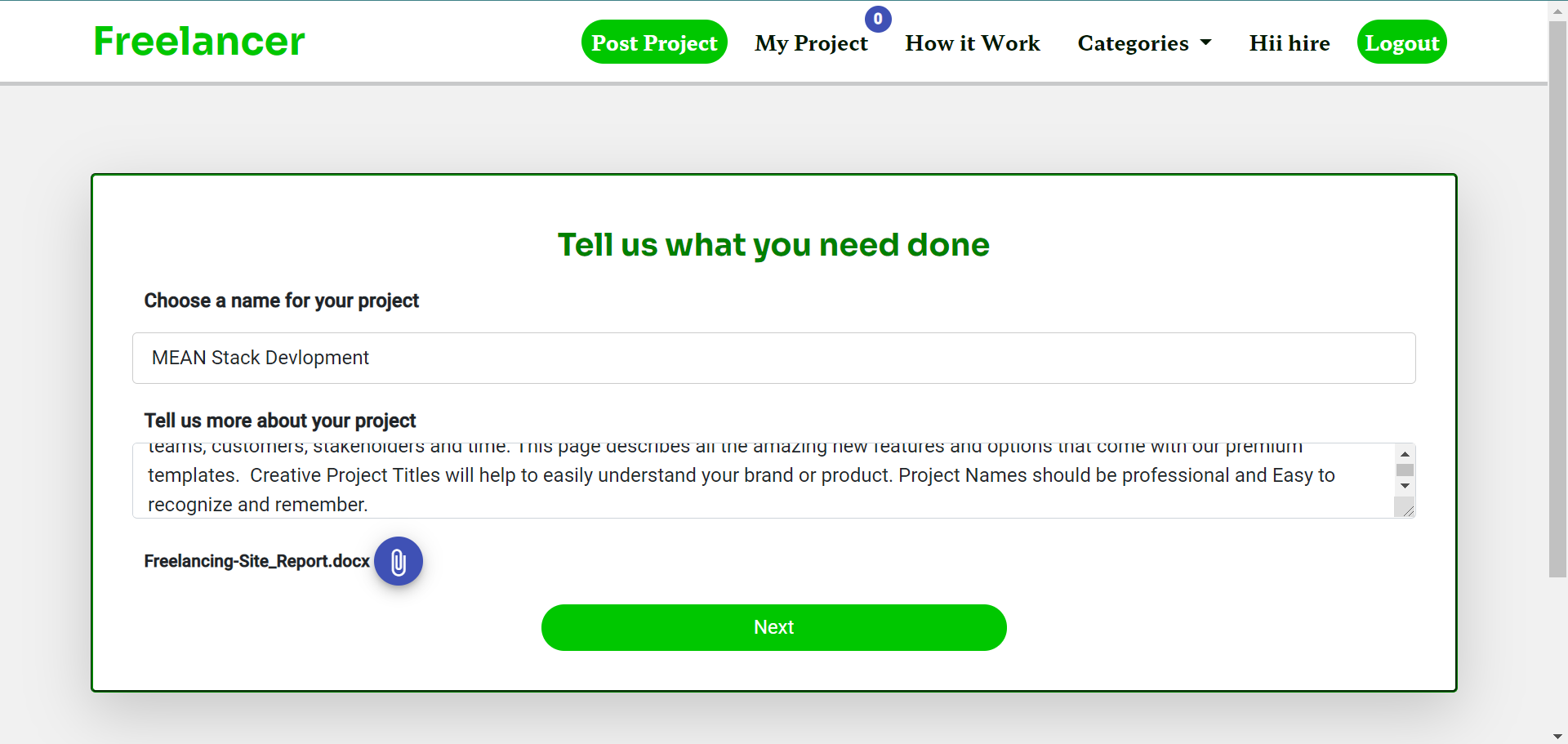
* Home:-

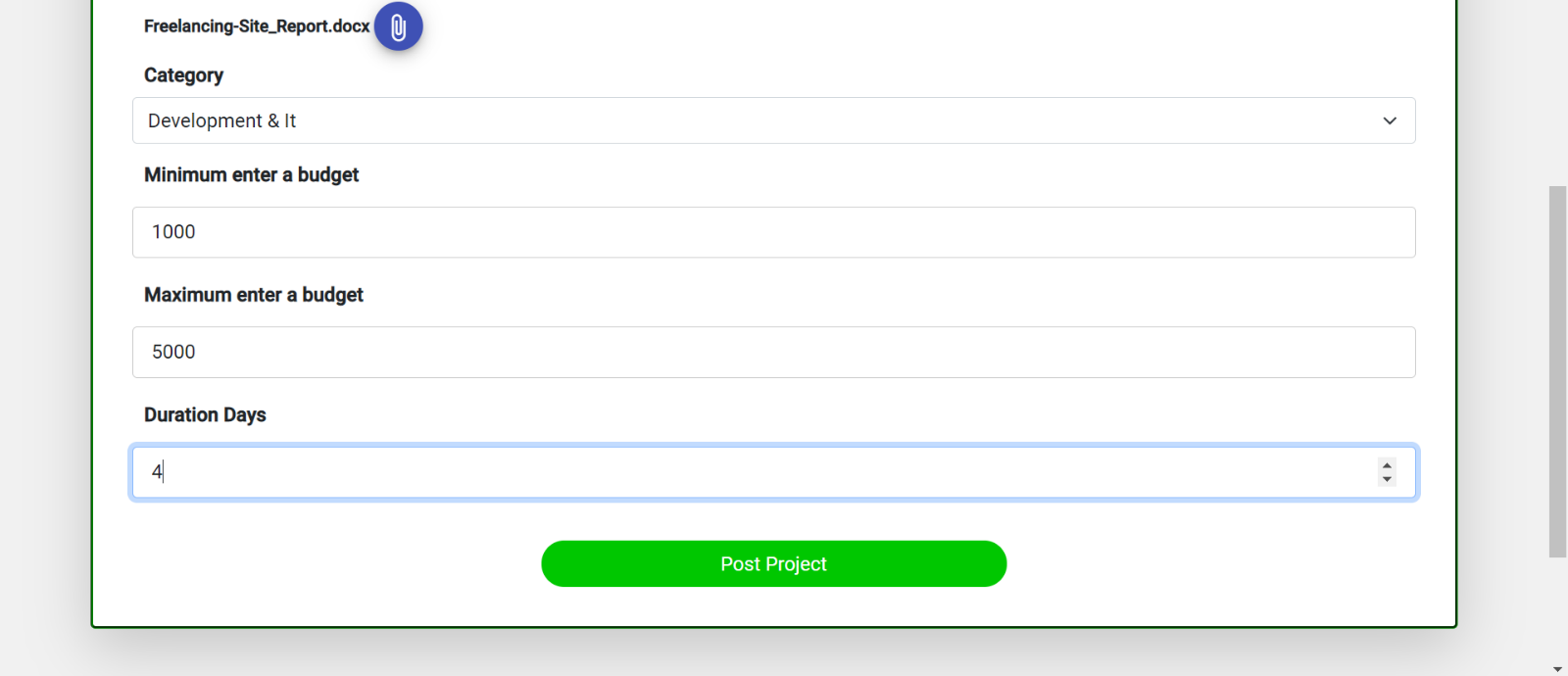


* How It Work:-

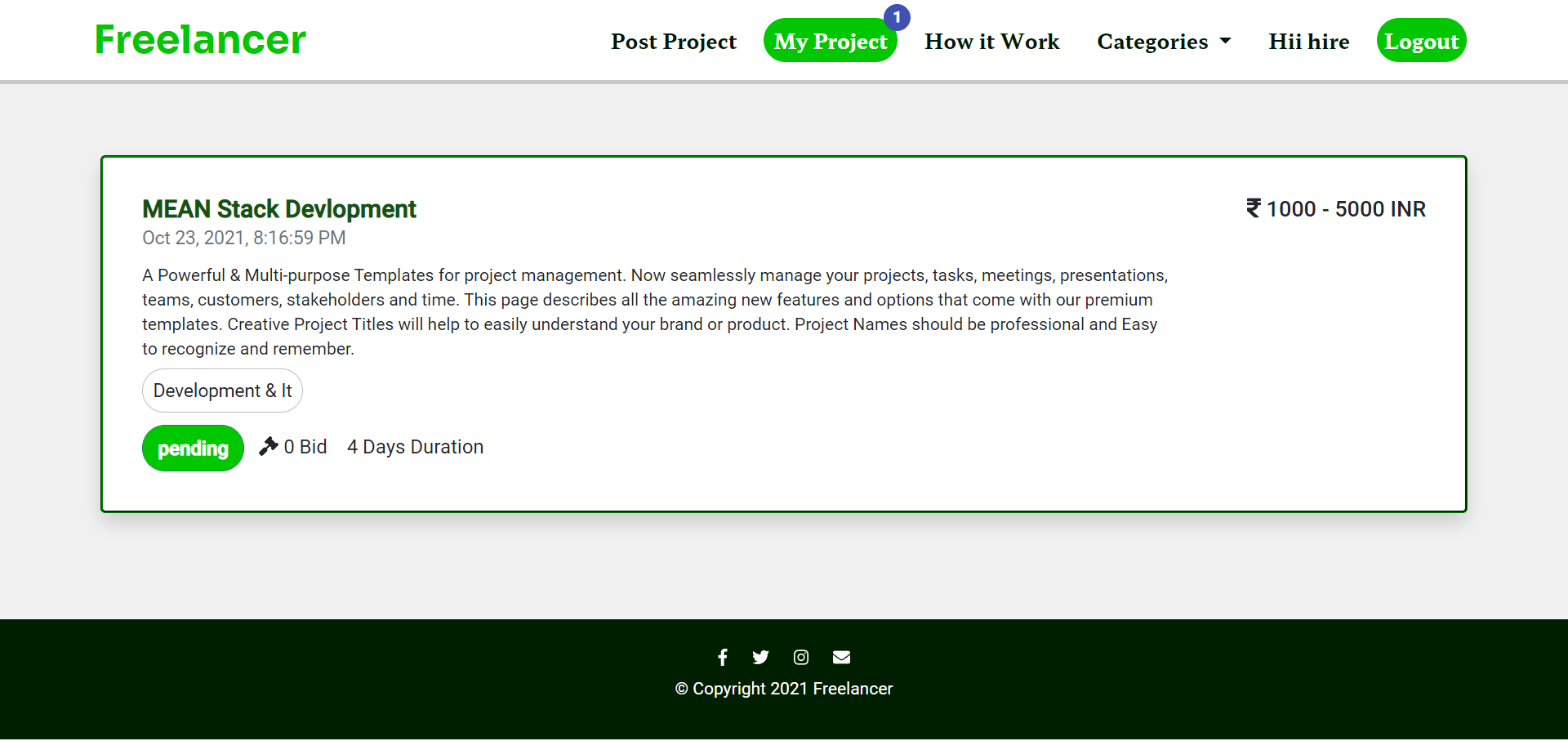


* Buyer Post Project :-

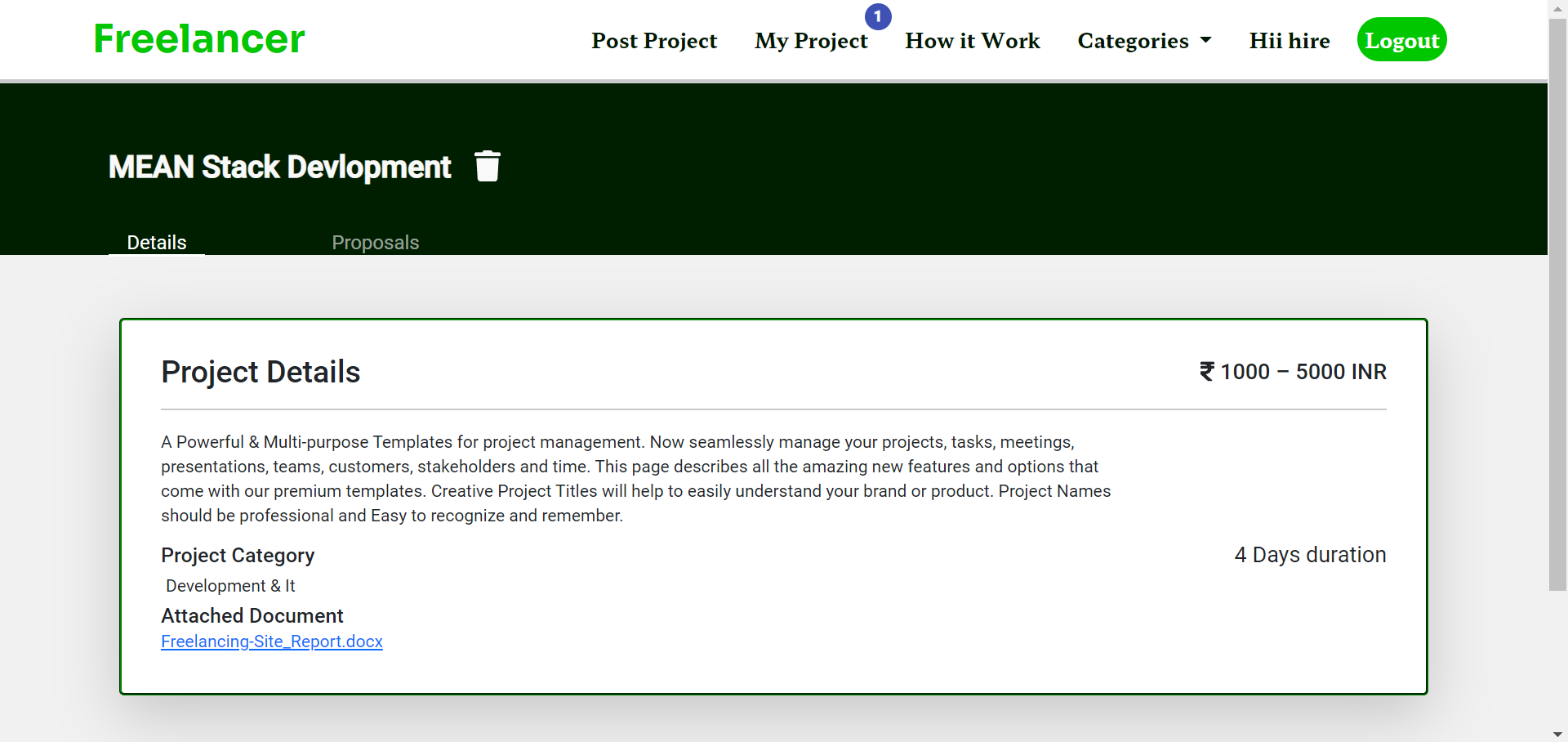




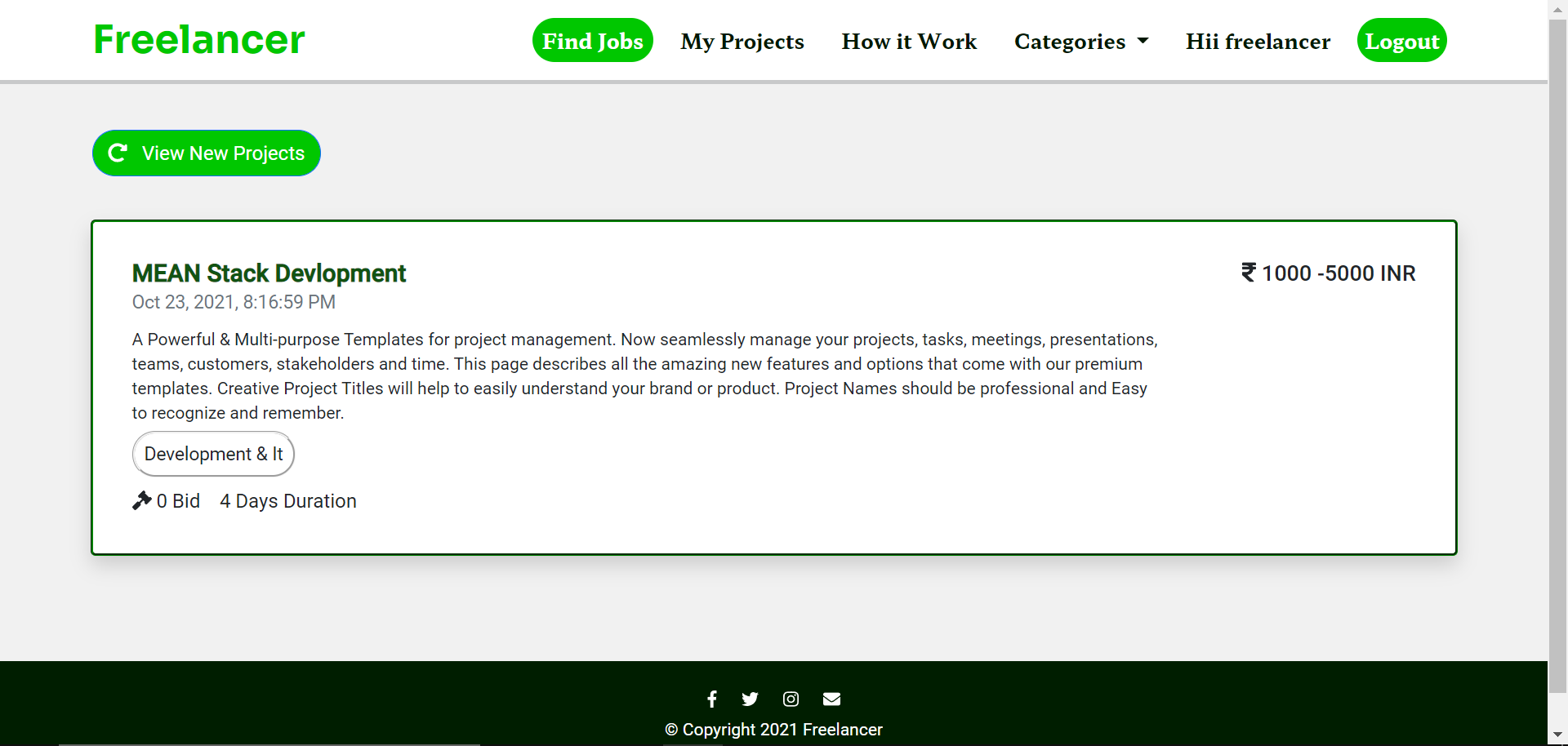
* Buyer’s My Project :-



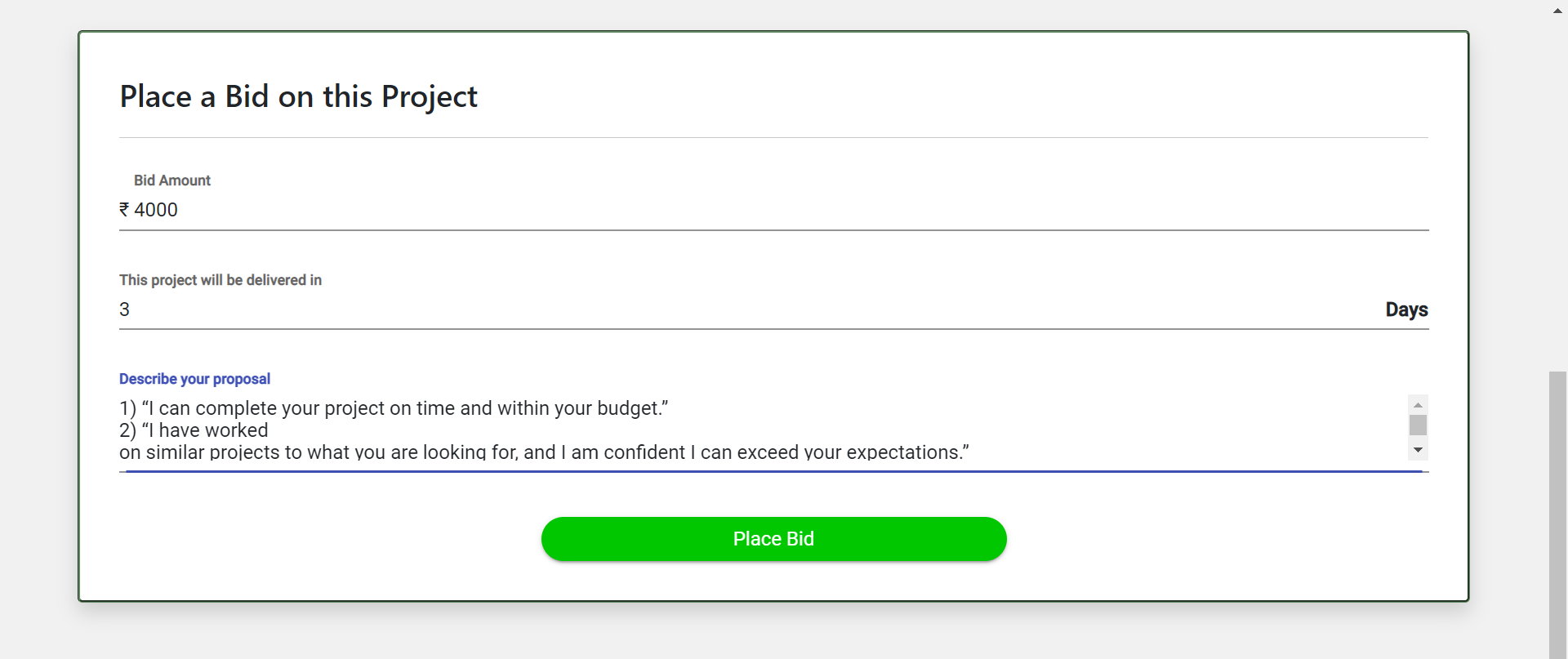
* Buyer’s My Project Details:-

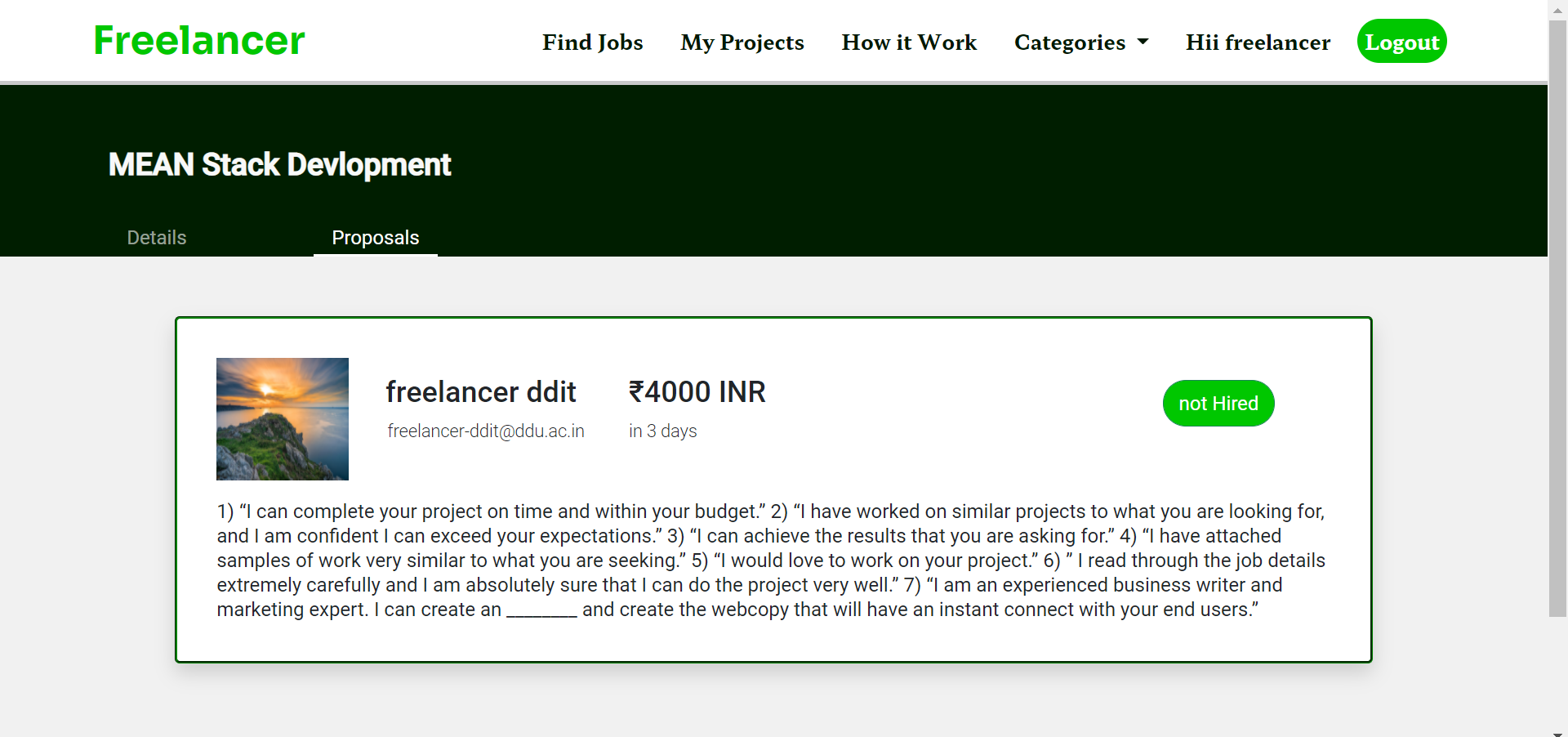


* Freelancer Find Jobs (Buyer Posted Project):-

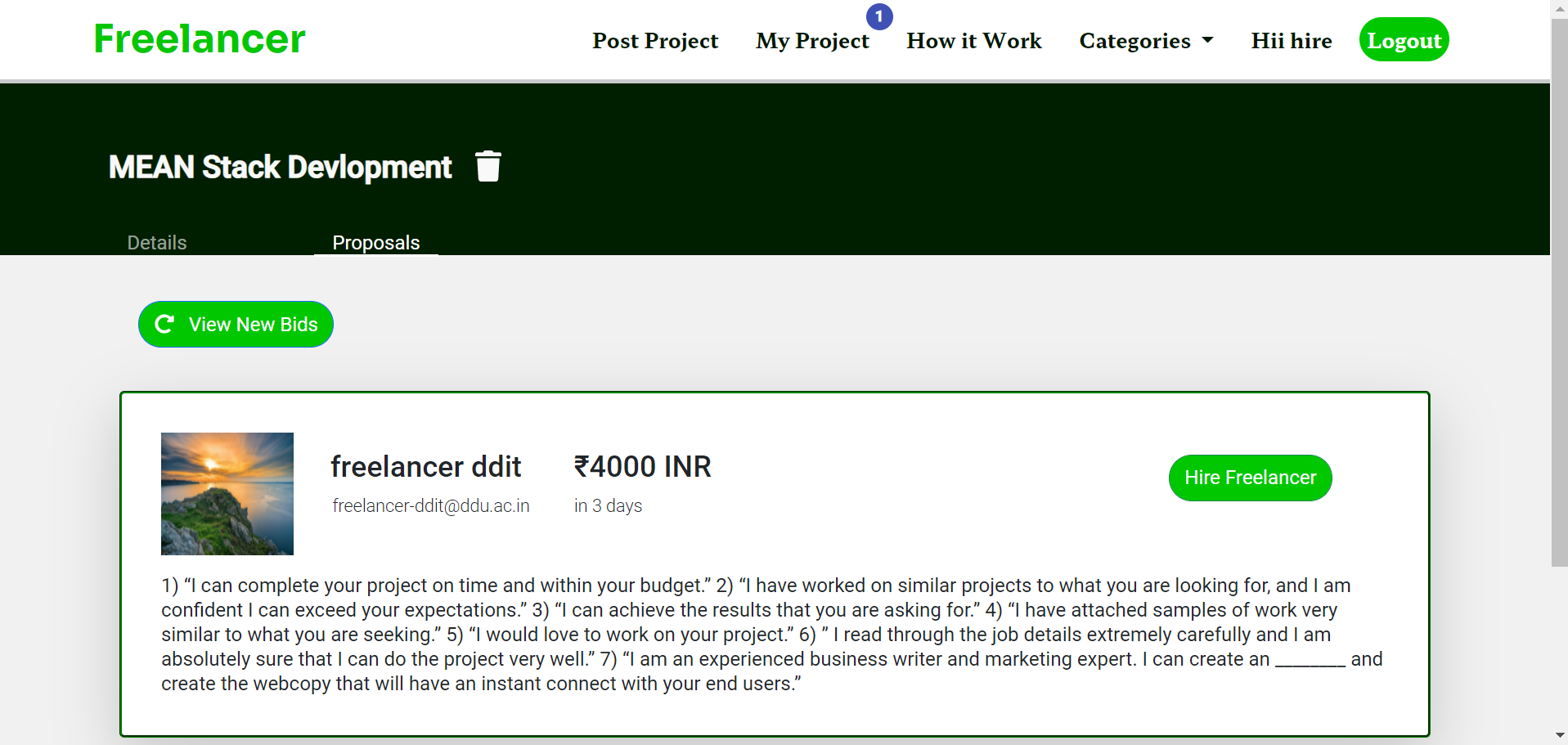


* Freelancer Place Bid:-

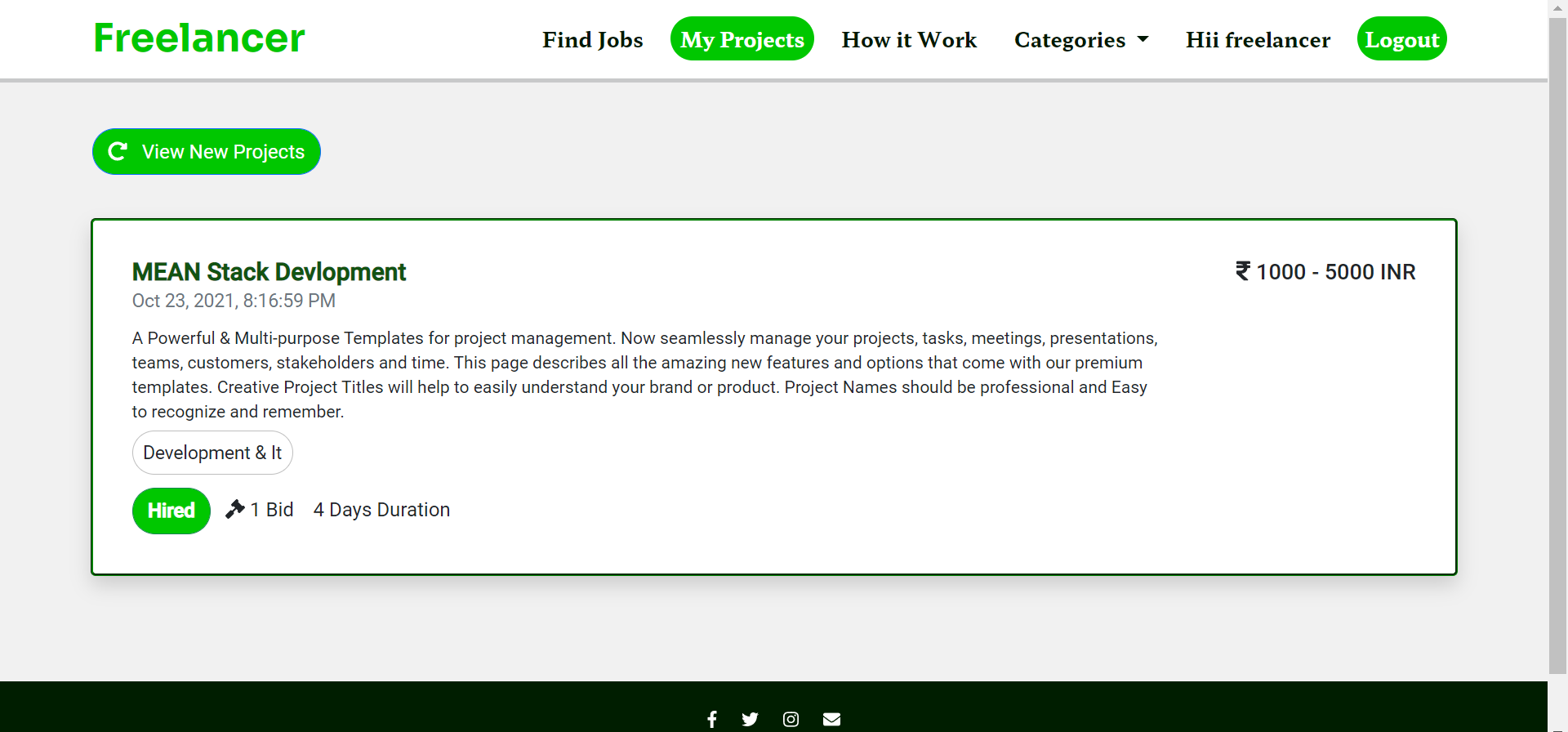




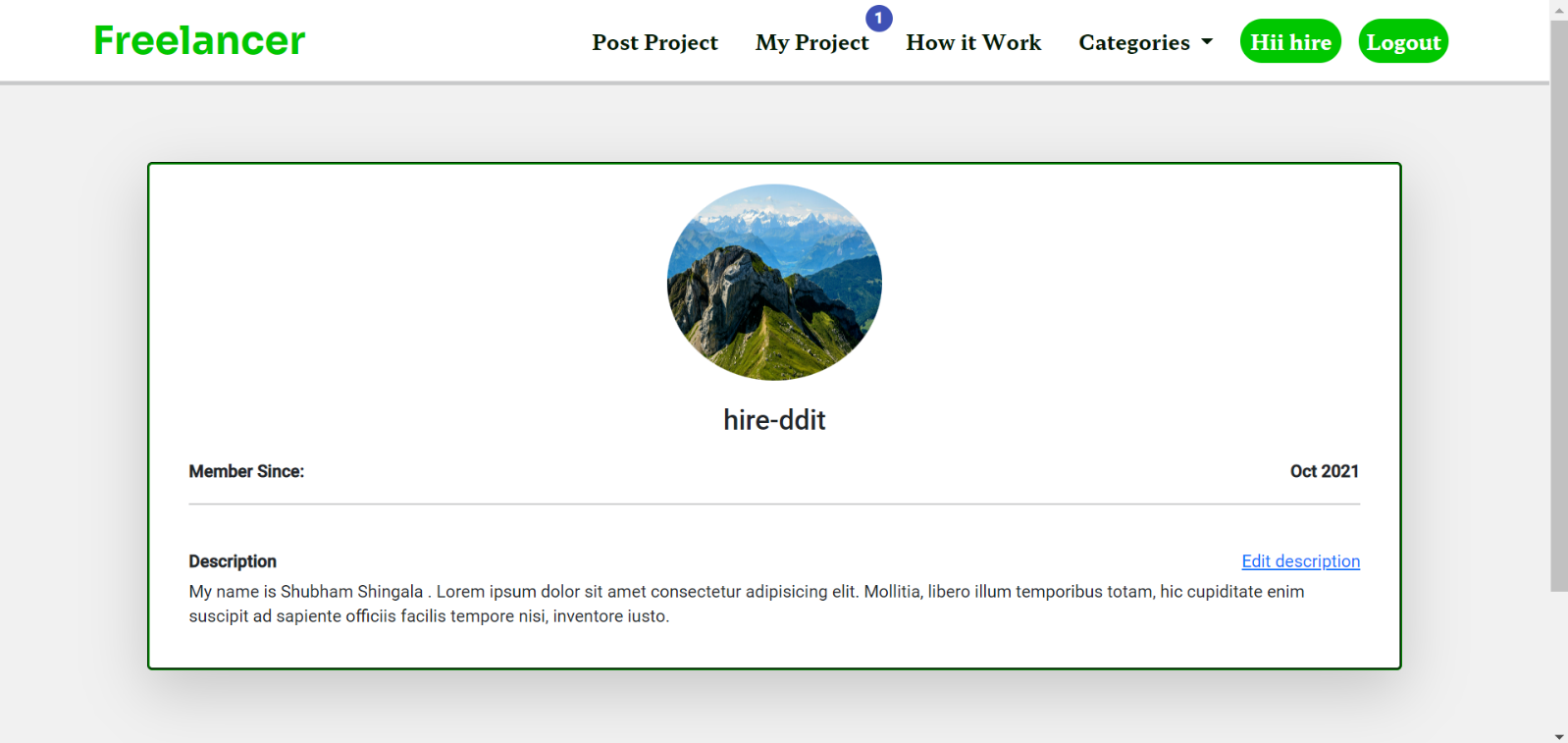
* Buyer Proposals For Their Posted Project:-



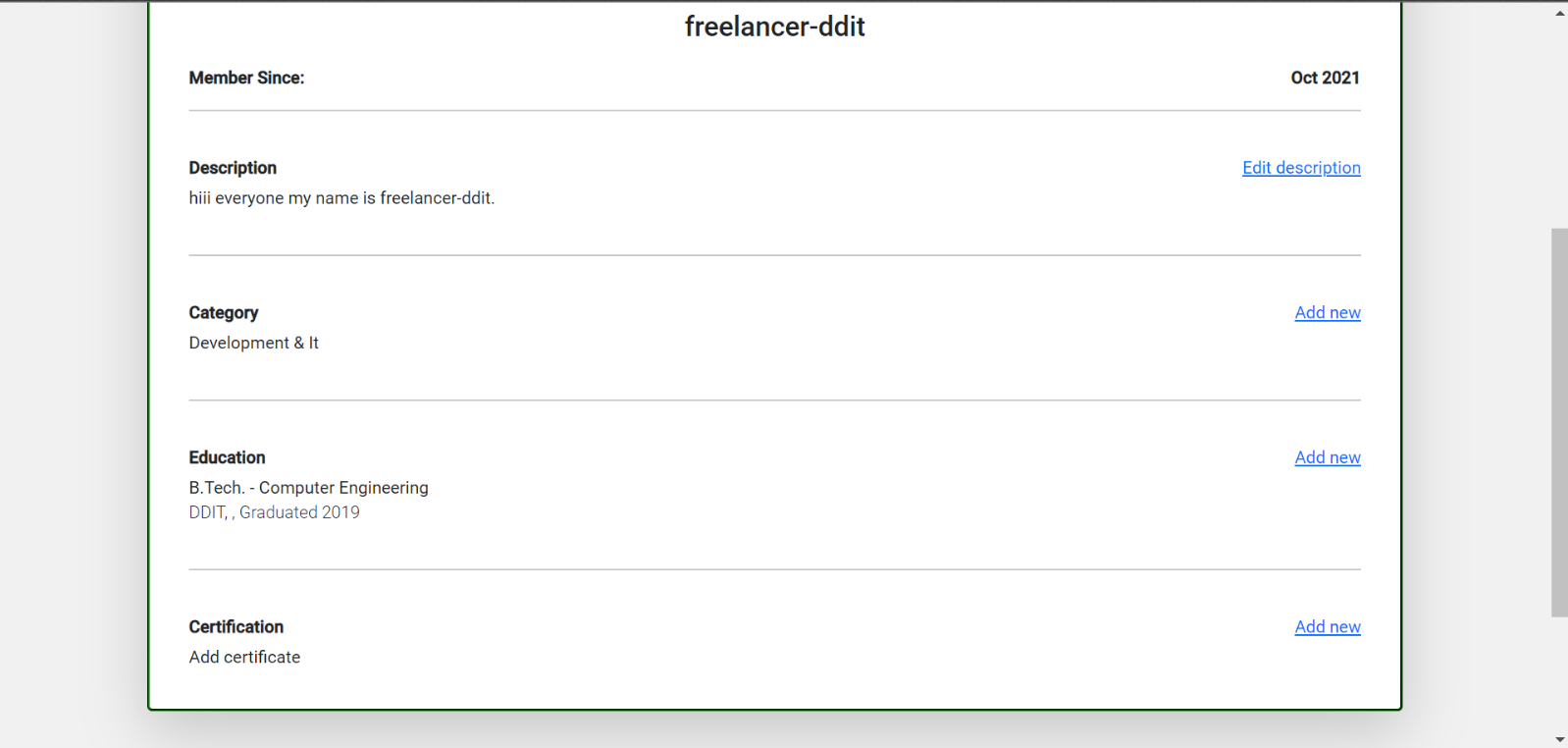
* Buyer My Project:-



* Buyer’s Profile:-



* Freelancer’s Profile:-



1. **Conclusion**

The functionalities are implemented in system after understanding all the

System modules according to the requirements. Functionalities that are

Successfully implemented in the system are :

* User Registration
* Login
* User authentication
* Logout
* Update Profile
* Post Project (Buyer)
* My Project (Buyer)
* Find Jobs (Freelancer)
* My Project (Freelancer)
* Update Status Of Project
* Place Bid (Freelancer)

1. **Limitations and Future Enhancements**

In future, we can add some functionalities like chat, video call between

buyer & freelancer to make their interactions more specific. We can add payment

gateway as well, we can add virus checker when someone upload the file to show

alert that file contains virus. also, we can add login using google or facebook as

well using their apis. We can add rating & reviews for both side like buyer &

freelancer so both side easy to place bids or hire freelancers.

1. **Reference / Bibliography**

Following links and websites were referred during the development of this

Project:

<https://stackoverflow.com/>

<https://github.com/>

<https://angular.io/>

<https://nodejs.org/>

<https://expressjs.com/>

<https://docs.mongodb.com/>