WILDLIFE TRACKER

Remotely controllable Real-time Efficient and effective tracking system

—DESIGN MANUAL—

TABLE OF CONTENTS

INTRODUCTION **2**
- Problem Overview 2
- Solution 2
- Features 3

HARDWARE DESIGN **4**
- Overall Design 4
  - CAD Design 4
  - Schematic Diagram of the whole Hardware System 6
- Control Unit 7
  - Components 7
  - Operations 10
  - Circuit Diagrams 11

WEB APPLICATION **13**
- Organization 13
- Front End 15
  - Technologies 15
  - User Interface 16
- Back End 25
  - Technologies 25
  - End Points 26
  - HTTP Response Status Codes 28
- Database 29
  - Database Schema 29
- Security 32

Testing **34**
- Unit Testing 34
  - Front-End Testing 34
  - Back-End Testing 35
- Integration Testing 36

Final Product **39**

INTRODUCTION

Problem Overview

The traditional method of wildlife research in Sri Lanka and many other developing as well as developed countries in the world are to stay in wild areas for a long time, observe animals randomly, and identify areas where animals are most active. This method is ineffective because some animals hide when they sense humans. They don't behave naturally around humans. Thus, it consumes a lot of time to observe their natural behaviors. Researchers have to stay in the wild throughout this long period. So, this is a formidable process. Sometimes their life is in danger because there is a possibility to face an attack by wild animals while they are in the jungle. An immense amount of time, money, and human resources are spent in vain on the traditional tracking of wild animals for researchers.

Solution

“The solution to this problem is a Remote controllable Real-time Efficient, and effective tracking system.”

This system consists of three major components, A hardware unit, a cloud server, and a web App. The hardware unit consists of camera traps, PIR sensors, and a location tracker. One unit of these three components makes a station. Researchers need to set up this station in the research area. One researcher can have more than one station established in different places according to their choice. The sensors in a station can detect animals when they are in the sensing range and it will trigger the camera. A real-time photo is captured at that moment and they are stored in the station itself and also sent to the cloud server. Then the researcher can analyze these observed data through the web app.

Features

- ☐ **Remotely Controllable**
  Researchers can monitor the station through the web app after it is successfully established in the research area.

- ☐ **High-Performance Camera Module**
  A powerful camera unit to capture perfect photos in any lighting condition. Also provides video recording facilities, and recorded videos are stored in the own memory of the station.

- ☐ **Tracking System**
  Pinpoint the exact location of the station. The most active station can be observed by the Web App.

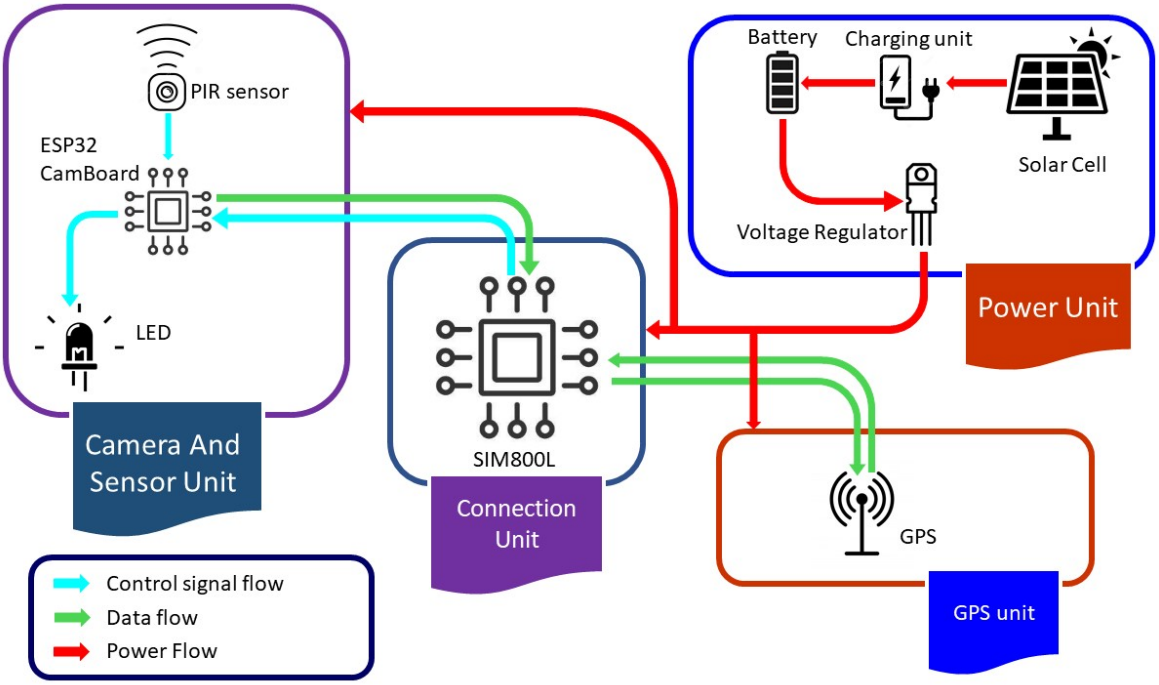
- ☐ **Solar Power**
  The stations are powered by batteries. A solar cell system is used to recharge them.

- ☐ **Durable**
  Can be set up in any environment. Made with highly durable and waterproof materials so that animals and unforgiving weather can't damage it.

- ☐ **Self Storage In Stations**
  Data obtained by the system is stored in itself when there is no connection with the database.

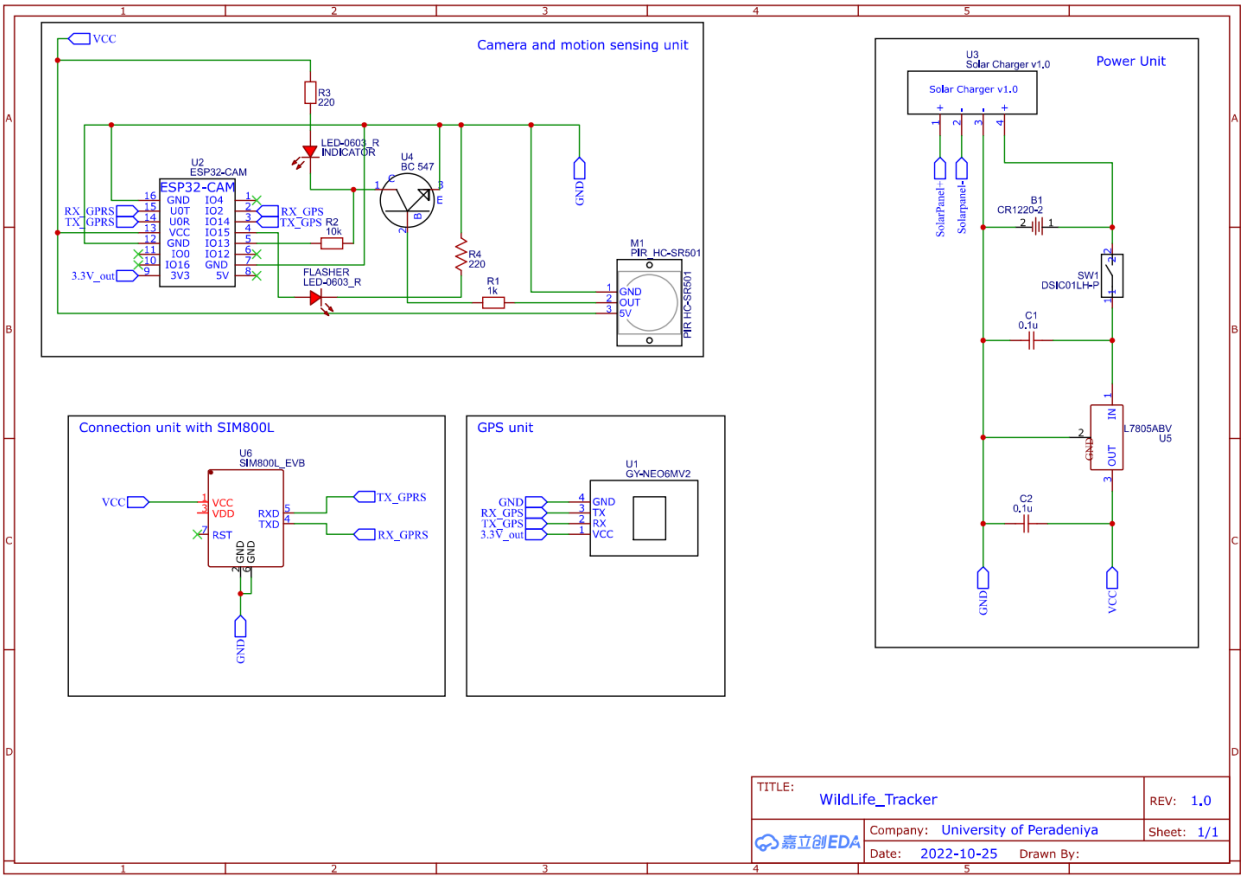
HARDWARE DESIGN

Overall Design



CAD Design

Schematic Diagram of the whole Hardware System

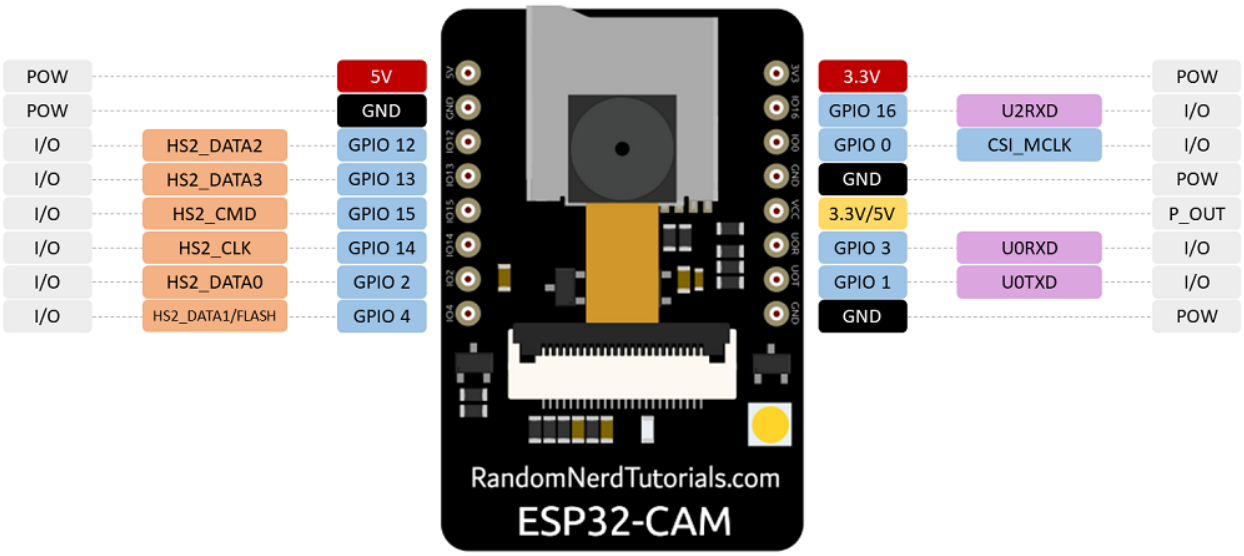


Control Unit

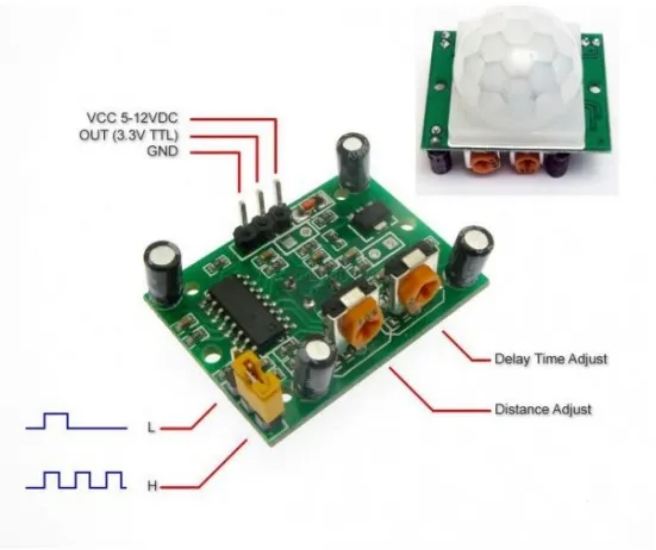
The main control unit of the wildlife tracker basically consists of the ESP32-cam module. This module handles several tasks/functionalities altogether. The main functionality of the overall hardware system is to get a photo of a wild animal when the PIR sensor gets triggered and the GPS location details and upload that data to the cloud database. After that, the software system takes care of the rest of the functionalities.

Components

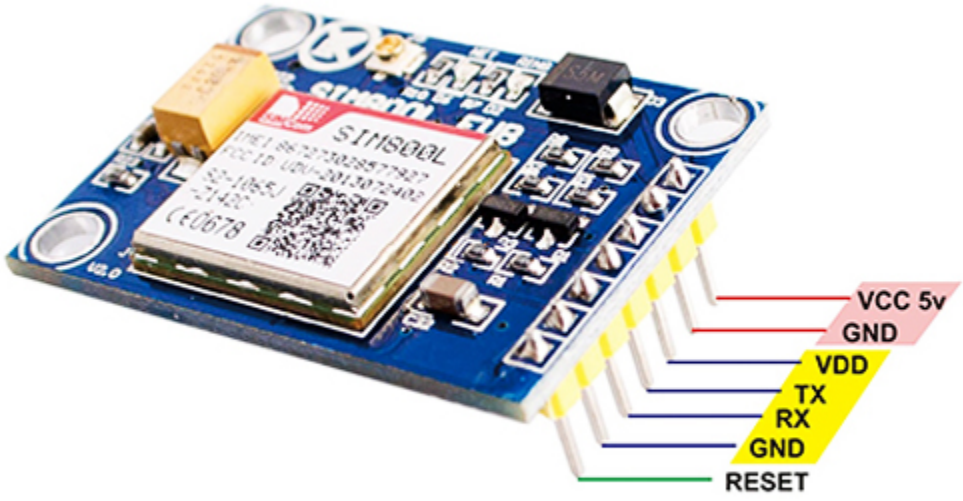
➔ ESP32 cam module



➔ PIR sensor unit and Flasher

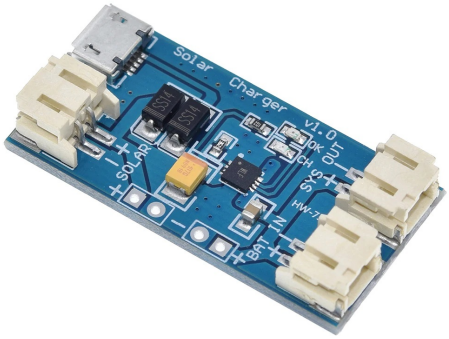


➔ SIM800i version2 module



➔ Neo6m GPS module

➔ Power unit



Operations

The main functionalities of the hardware units

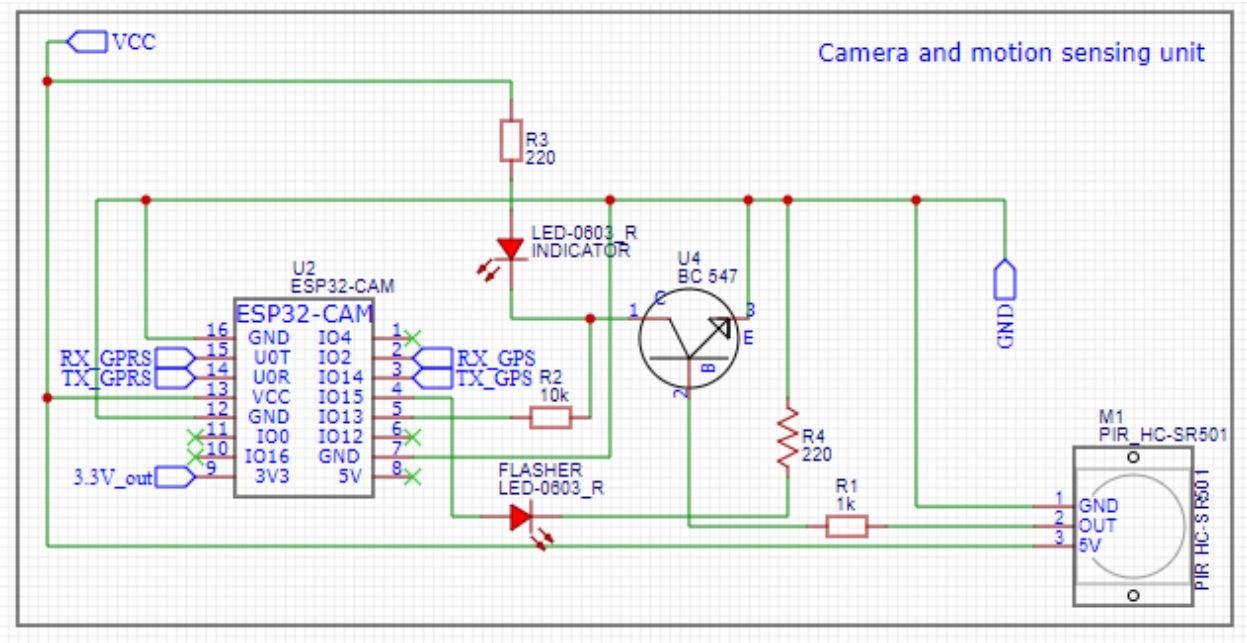
- ➔ ESP32 Cam module
  - ◆ Acts as the central processing unit of the whole hardware system.
  - ◆ Takes a photograph by the control data from the PIR sensor.
  - ◆ Does the UART communication in between the units.
  - ◆ Inbuilt SD card reader to store all the photographs taken by the camera when the connection is disrupted with the server.

- ➔ PIR sensor unit and Flasher
  - ◆ A custom-designed circuitry to sense data in the environment and provide the controlling data back to the camera module.
  - ◆ An Passive Infrared sensor is used to sense an animal in the nearby environment.
  - ◆ A bright LED is used as a flasher to get the photographs at low light levels. This provides an extra advantage.

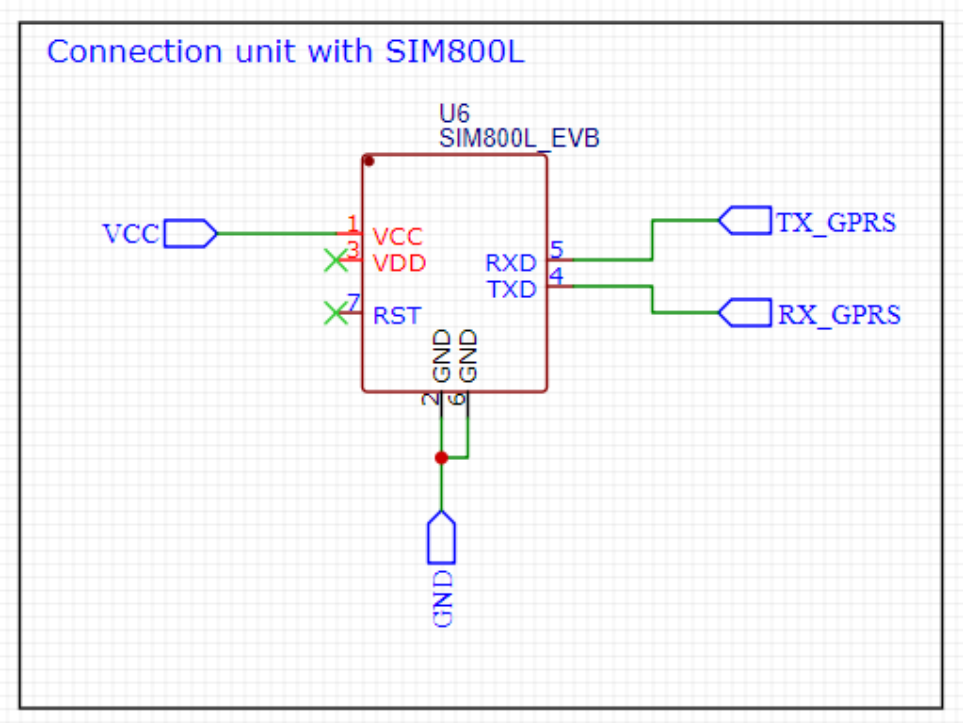
- ➔ Sim800l version2 module
  - ◆ This module handles the communication between the cloud server and the hardware unit.
  - ◆ Uses a 2G signal to communicate with the server.
  - ◆ Very low power-consuming module.

- ➔ Power unit
  - ◆ This unit made with custom-designed circuitry powers up all the modules described above with the required power settings.
  - ◆ It is equipped with an additional solar panel which will charge the battery unit.
  - ◆ Very ideal for harsh and rural environments.

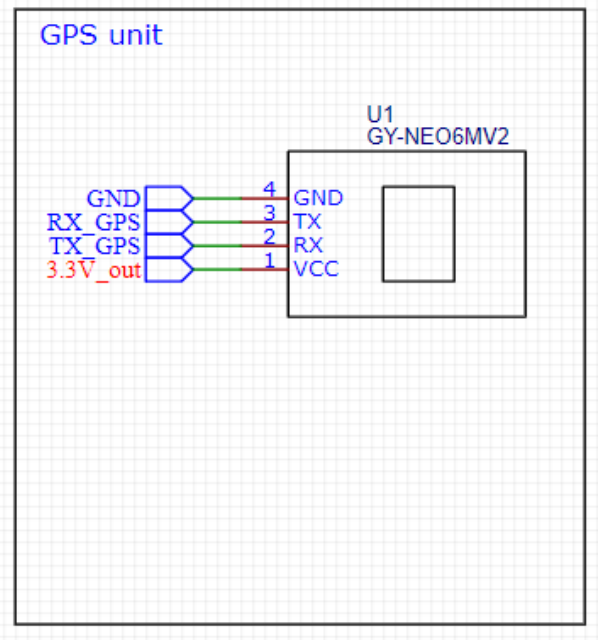
Circuit Diagrams



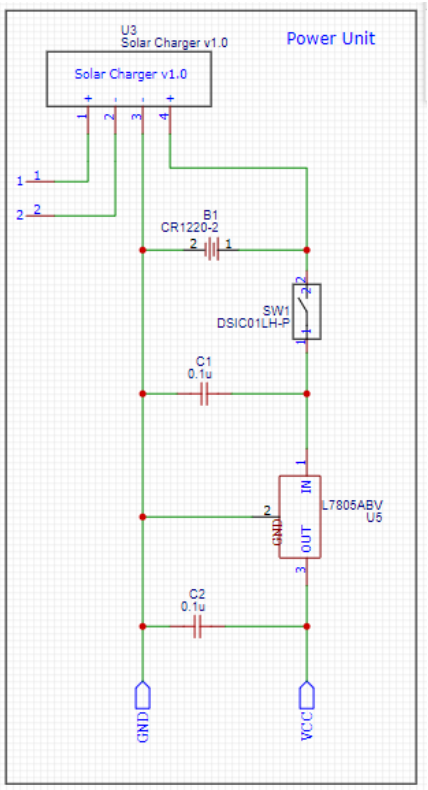
Camera and Motion Sensing unit Schematic Diagram



Sim800l Module schematic diagram



GPS Module Schematic Diagram



Power unit schematic Diagram

WEB APPLICATION

The web application was built using the MERN stack.

- MongoDB - Database
- Nodejs & Express - Backend
- Reactjs - Frontend

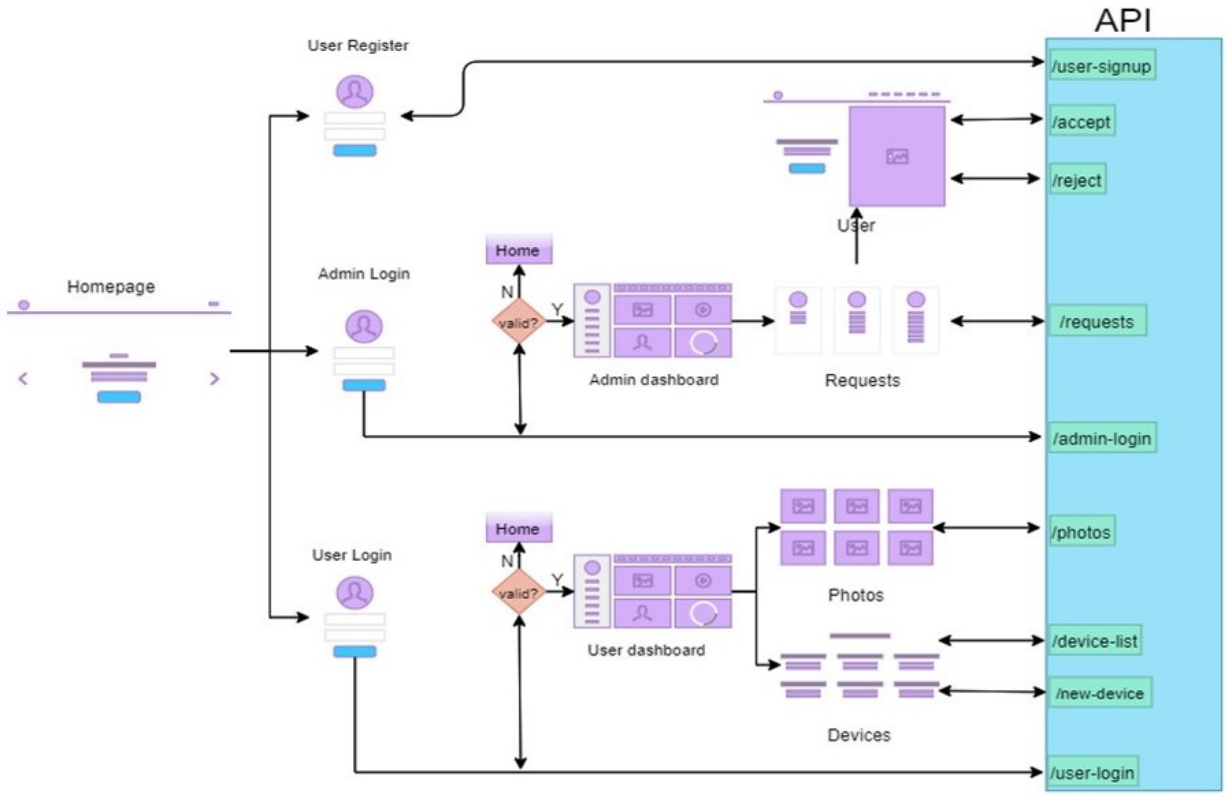
Web Application is used by Admins and Users.

- Admins
  - Accept/Reject user registration requests.
  - Add new admins to the system.

- Users
  - Connect hardware devices to the system.
  - Check photos taken by devices.

Once a user has registered in the system, he has to wait for the acceptance of an admin. Admin should check registration requests made by different clients. Before accepting the request, they should verify that the request has valid and true information. After successful registration user can log in to the system. Then, there is an option to connect their devices to the system. After that, the photos tab in the web application will show every photo taken by devices owned by the user.

Organization



Front End

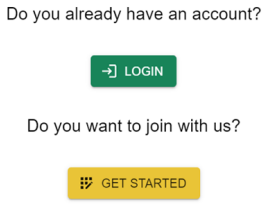
Technologies

Library	Version	Description
react	17.0.2	
react-dom	17.0.2	
react-router-dom	5.2.0	
react-scripts	4.0.3	
axios	0.21.1	For HTTP requests
list-react-files	0.2.0	To handle files
react-animated-text	0.1.4	Text animations
react-bootstrap	1.6.1	Boostrap library
react-helmet	6.1.0	To change page headers
react-icons	4.2.0	Icons library
react-leaflet	3.2.1	To show animals on a map
react-loader-spinner	4.0.0	Loading animations
react-modal	3.14.3	Pop-up windows
react-phone-input-2	2.14.0	To style phone number input field
react-pro-sidebar	0.6.0	Sidebar library
react-reveal	1.2.2	Scroll animations
react-slideshow-image	3.5.1	To make image slideshows
semantic-ui-css	2.5.0	Styling

semantic-ui-react	2.1.3	Styling
universal-cookie	4.0.4	To handle browser cookies

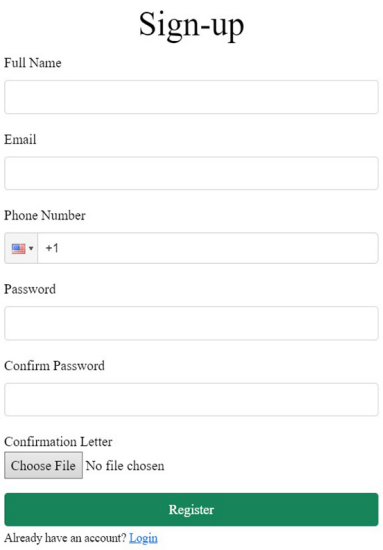
User Interface

1. Home



This is the home page of our web application. Clients can log in to their accounts by clicking on the log-in button. The get started button directs clients to the user registration form. The button provided in the right bottom corner provides a link to contact us via email.

2. Register

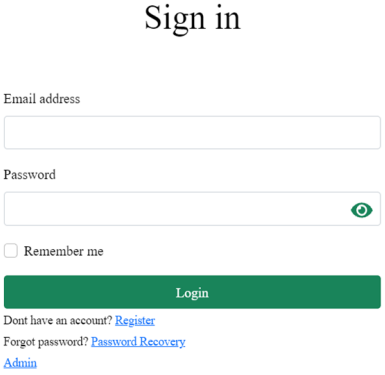


Here is the two registration pages in our web application.

01. The user registration Page
02. The admin registration Page

Form validations and feedback are used to give a better user experience. Clients have to fill all the fields correctly in the forms to enable the submit button. The button provided in the left top corner directs the user to the home page.

3. Login

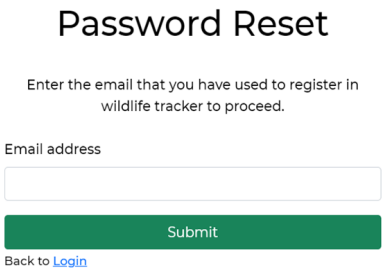


Here are the two login pages in our web application.

01. The user login Page
02. The admin Page
02. The error message if the credentials are not valid.

Form validations and feedback are used to give a better user experience. The user has to provide the correct credential to log in to the system. If the credentials are not matching it will display the error message. With an error message, links back to the login page or password recovery are provided. Also in the login pages, we have provided a link to password recovery. Users can reset their password using this link.

4. Password Recovery



There are two pages for the password recovery process. The first one is for requesting password recovery. Here, users have to enter the email of their account and submit it. If the entered email is valid, it will show up the success message. Else, it will show up the error message.

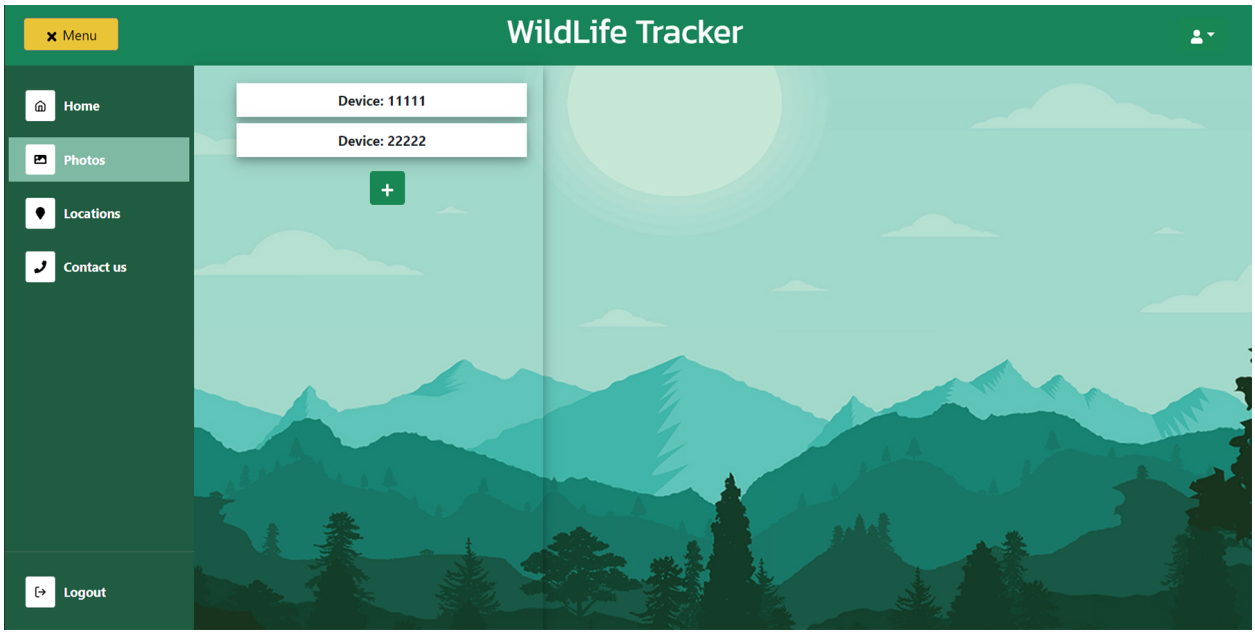
5. User Dashboard

These are the pages in the user dashboard.
There is a home tab, a photos tab, a location tab, and a contact tab.
In the photos-tab user can see the photos taken by devices connected to his profile. Also, there is an option to connect new devices with the profile.
The map provided in the location tab pinpoints all the locations of devices connected to the user's profile.

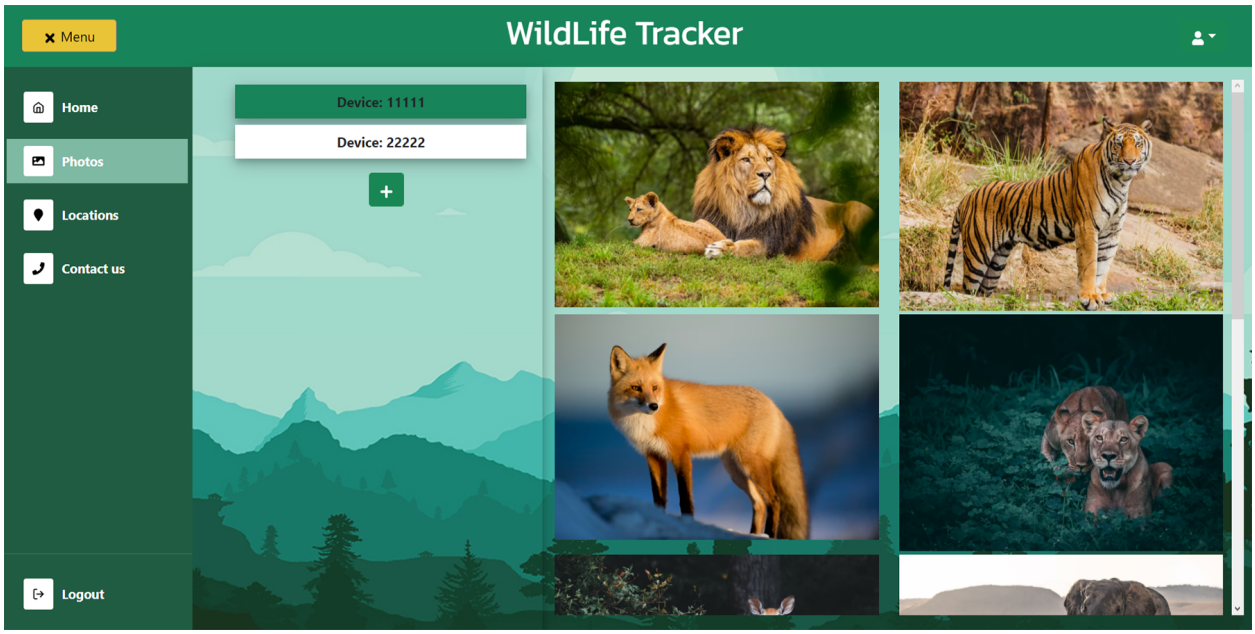
Home Tab

Photos Tab

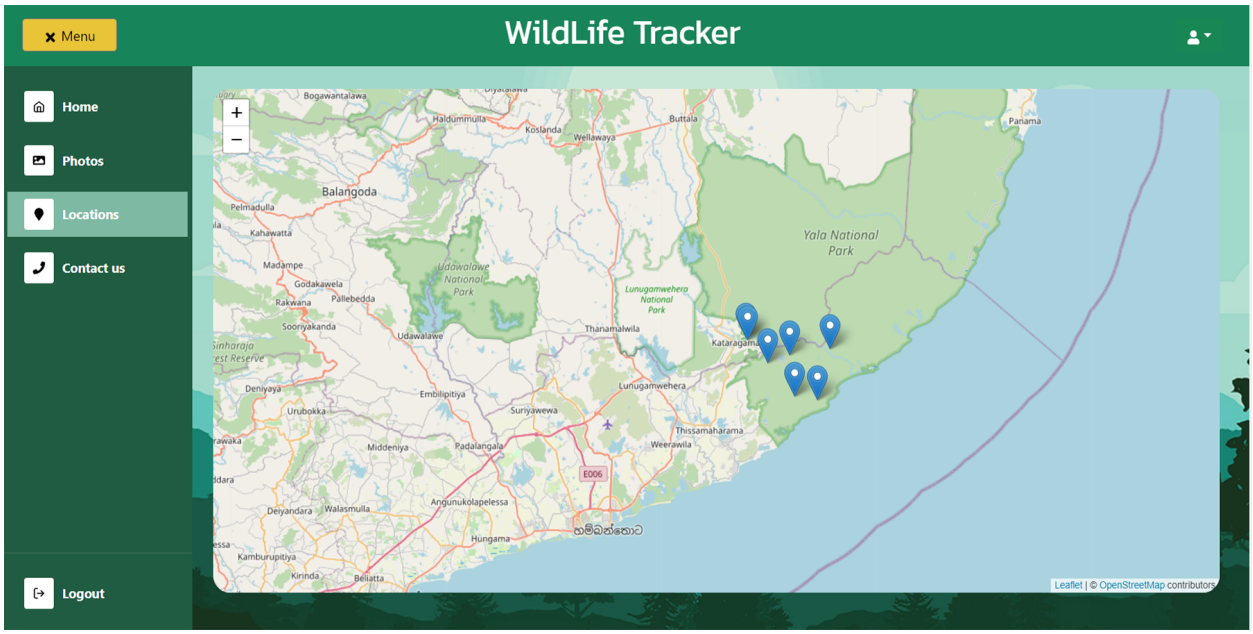
1. Devices



2. photos

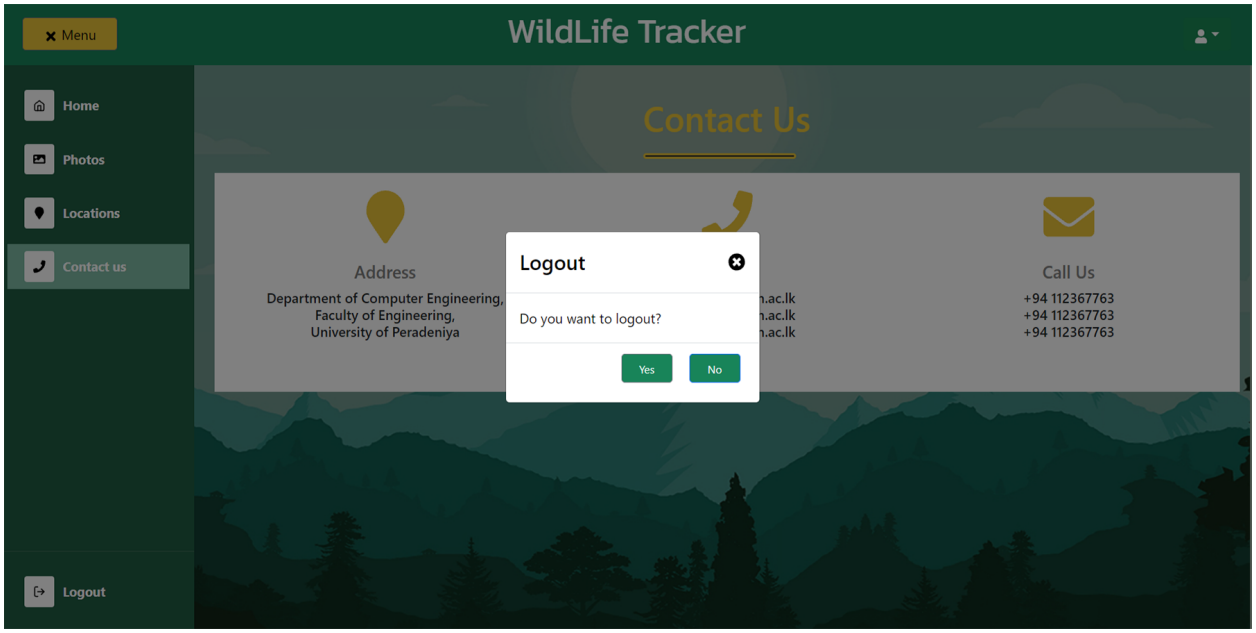


Location Tab



Contact Us Tab

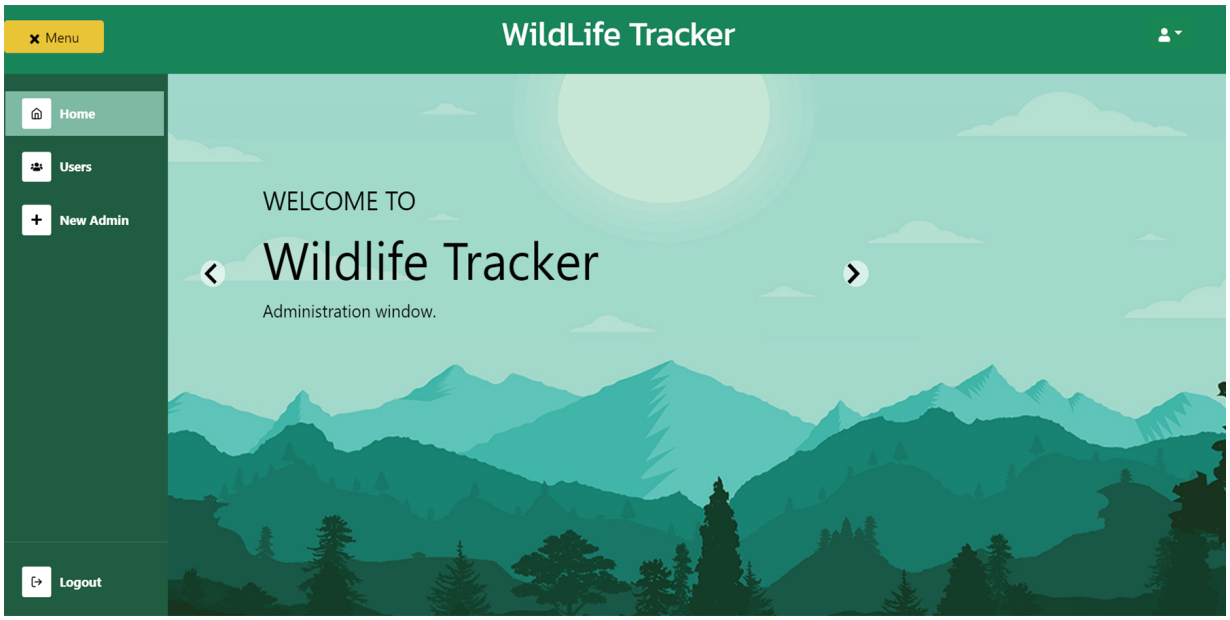
Logout Tab



6. Admin Dashboard

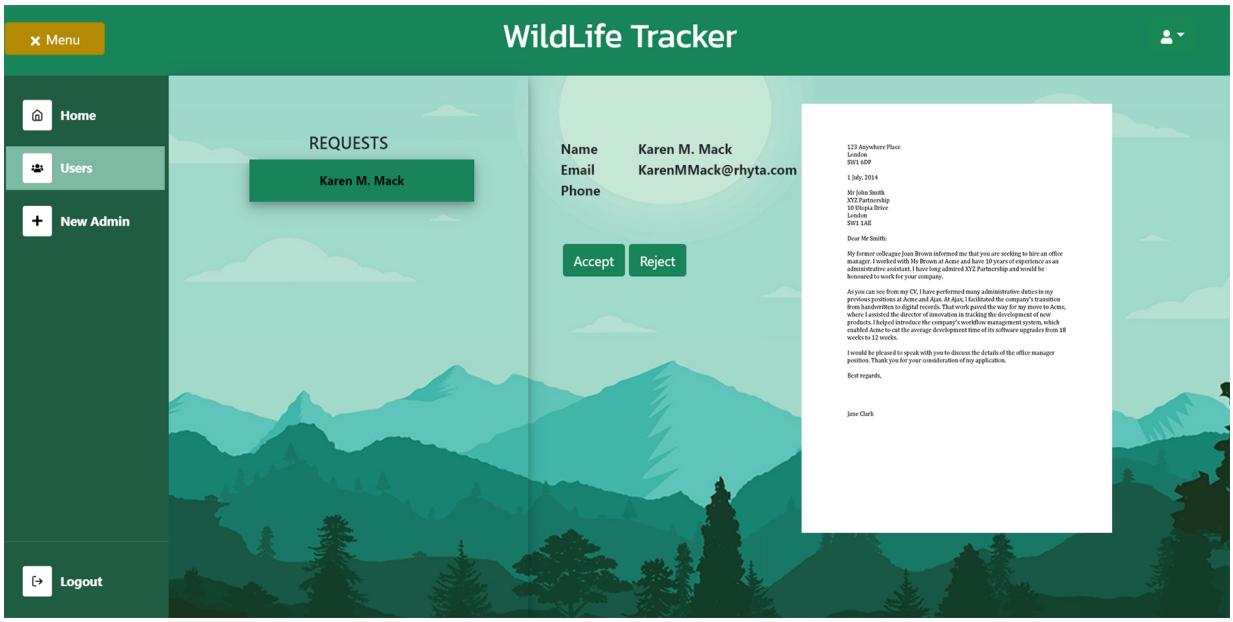
Here are the pages in the admin's dashboard. The user tab will show all the user requests to the system.
Admin can see details and the letter uploaded by each user by clicking on a request in the list. Then they can submit or reject the request. If they are rejecting the request, they have to provide the reason for the rejection in the given text box.

Home Tab

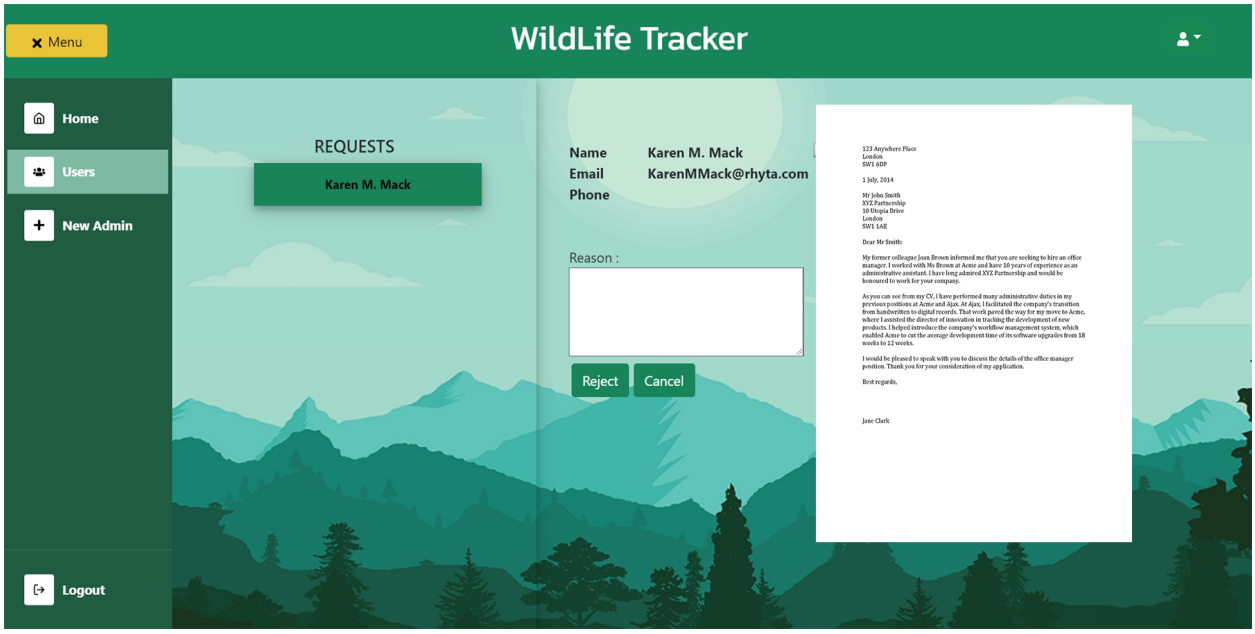


Users Tab

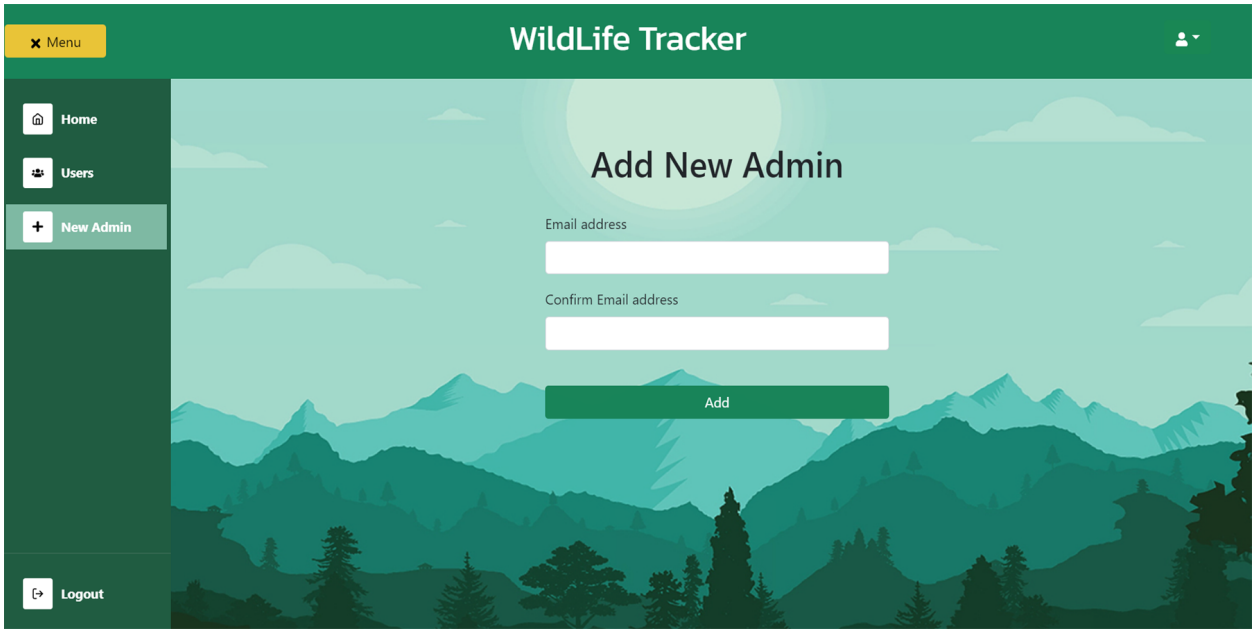
1. Registration Requests



2. Rejecting a registration request



New Admin Tab



Back End

Technologies

Library	Version	Discription
express	4.17.1	To develop REST api
cors	2.8.5	To enable cross-origin sharing
jsonwebtoken	8.5.1	JWT libarary
mongoose	5.13.3	To careate a connection with MongoDB databse
dotenv	10.0.0	To load environment variables from .env file
fs	0.0.1-security	To work with file system
body-parser	1.19.0	To parse body as a string
bcrypt	5.0.1	For password hashing
nodemailer	6.6.3	To send emails

End Points

HTPP requests from front end to back end should be sent to following endpoints.

/signin-admin
- To log in as an admin to the system.
- Request body should have the email and the password.

/signin-user
- To log in a user to the system.
- Request body should have the email and the password.

/user-password-reset-rq
- To send a password reset request for a user.
- Request body should have the email of the user.

/admin-password-reset-rq
- To send a password reset request for an admin.
- Request body should have the email of the admin.

/password-reset/:email/:token
- To replace the previous password with a new password.
- Request body should have a new password and email of the user.

/register-admin/:email/:token
- To register a new admin
- The request body should have every registration information.

/register-user
- To register a new user
- The request body should have every registration information.

/accept-reg
- To accept a particular registration request
- The request body should have the email of the user who need to be registered.

/add-admin

- To add a new admin
- The request body should have the email of the admin who need to be registered.

/reject-reg

- To reject a particular registration request
- The request body should have the reason for rejection, and the email of the user who needs to be rejected.

/connect-device

- To connect a new device with the system
- The request body should have the serial number and secret code of the device..

/devices_list

- To get all devices connected to the system.
- Request body should have the email of the user.

/device_photos/:deviceID

- To get all photos captured by a particular device.
- Request body should have the device Id.

/delete_photo/:device&:photoURL

- To delete a particular image
- The request body should have the device id and the user of the photo.

/device_location/:deviceID

- To get the location of a particular device
- The request body should have the device Id.

/delete_user

- To delete a user
- The request body should have the email of the user to be deleted.

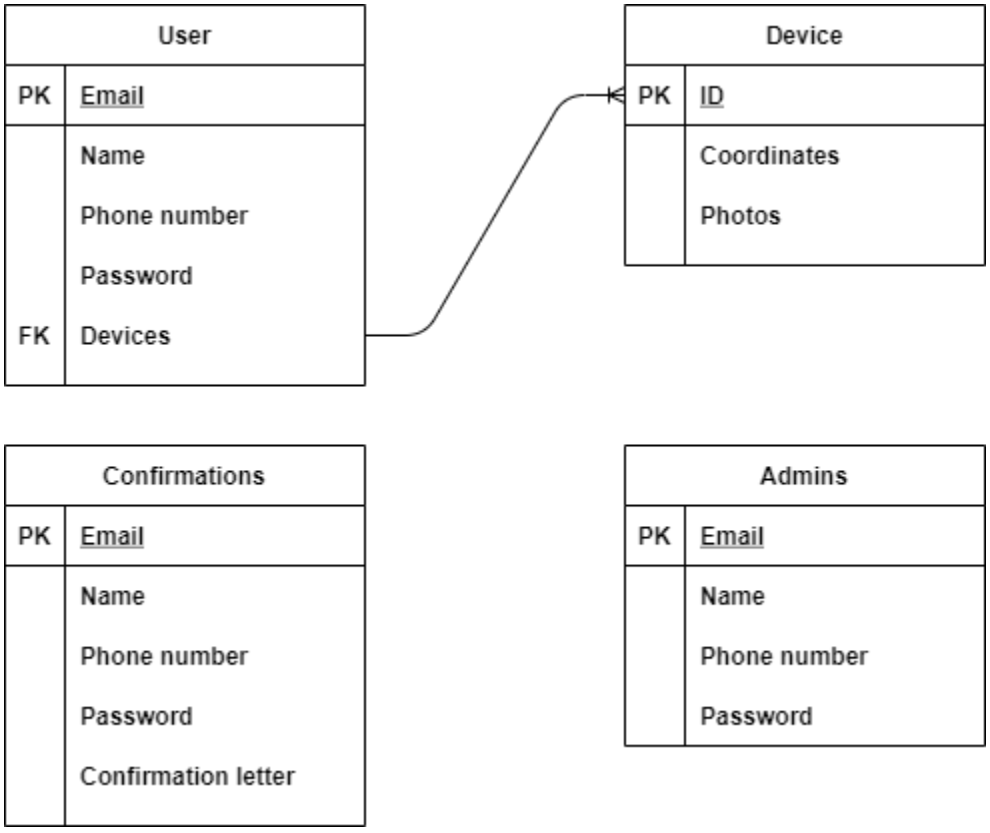
HTTP Response Status Codes

Code	Status	Usages
200	Successful	Successfuly log-in Device is connected successfully Registration request is successfully rejected
201	Successful	Password reset email is sent Password is successfully reset Registration acceptance is successful
401	Client errors	Invalid password Invalid email Registration link is invalid or expired Registration acceptance is failed No photos found
403	Client errors	Incorrect password of device Every unexpected error
404	Client errors	Device id is not found
409	Client errors	Email is already existing
422	Client errors	Validation errors in input fields
500	Server errors	Password reset is failed Registration request rejection is failed

Database

We use MongoDB as our database management system. Because of its scalability, powerful querying capabilities, and flexibility, it provides for modeling and manipulating data structures.

Database Schema



User Schema

```
let userSchema = new Schema(
    {
        name: {
            type: String,
            required: true
        },
        email: {
            type: String,
            unique: true,
            required: true
        },
        phonenumber:{
            type:Number,
            required: true
        },
        password: {
            type: String,
            required: true
        },
        devices: {
            type: [Number]
        }
    },
    {
        collection: 'users'     //name the collection
    }
)
```

Admin Schema

```
let adminSchema = new Schema(
    {
        name: {
            type: String,
            required: true
        },
        email: {
            type: String,
            unique: true,
            required: true
        },
        password: {
            type: String,
            required: true
        }
    },
    {
        collection: 'admins'     //name the collection
    }
)
```

Security

Our system handles really sensitive information about wildlife. Therefore, the CIA triad (Confidentiality, Availability, Integrity) of the application is maintained in the best way possible.

1. JsonWebTokens (JWT) are used for the authorization of users. These tokens are stateless, portable, high performing and decentralized which improves the security and scalability of the web application as well.

    - Once a user has successfully logged in to the system, the back-end should generate a JWT token, and attach it with the HTTP response.

```
let jwtToken = jwt.sign({
    email: getUser.email,
    userId: getUser._id,
    isAdmin : true
},"longer-secret-is-better",{
    expiresIn: '2h'
})

res.status(200).json({
    token: jwtToken,        //c
    expiresIn: 6600,        //c
    msg: getUser
})
```

    - Every HTTP request that comes from the client side should have included this JWT token in the header, Otherwise, the back-end won't provide the requested information as the response.
      Token can be stored in the cookies.

```
axios.get(`${URL}api/auth/devices_list`, {
    headers: {
        'x-auth-token' : cookies.get('token')
    }
})
```

- Also, the back end should verify that the JWT token is valid and not expired.

```
try{
    const token = req.header('x-auth-token')
    //console.log(token)
    jwt.verify(token, "longer-secret-is-better")
    const decodedToken = jwt.decode(token);
    req.userEmail = decodedToken.email;
    next()                                    //exe
}catch(error){
    res.status(401).json({
        message: "No token. Cannot Authorize",
        error: error
    })
}
```

2. HTTPS protocol is used for secure communication.

3. A special system of user registration is introduced to ensure the users of our web app are researchers.

Testing

Unit Testing

Front-End Testing

As a unit test we tested the functionality of the front end of our web application. This is very important because the front end is used by our clients. So, it must provide all the expected outcomes correctly to give a better user experience. Therefore, we needed to make sure that the front end of our web application works fine.

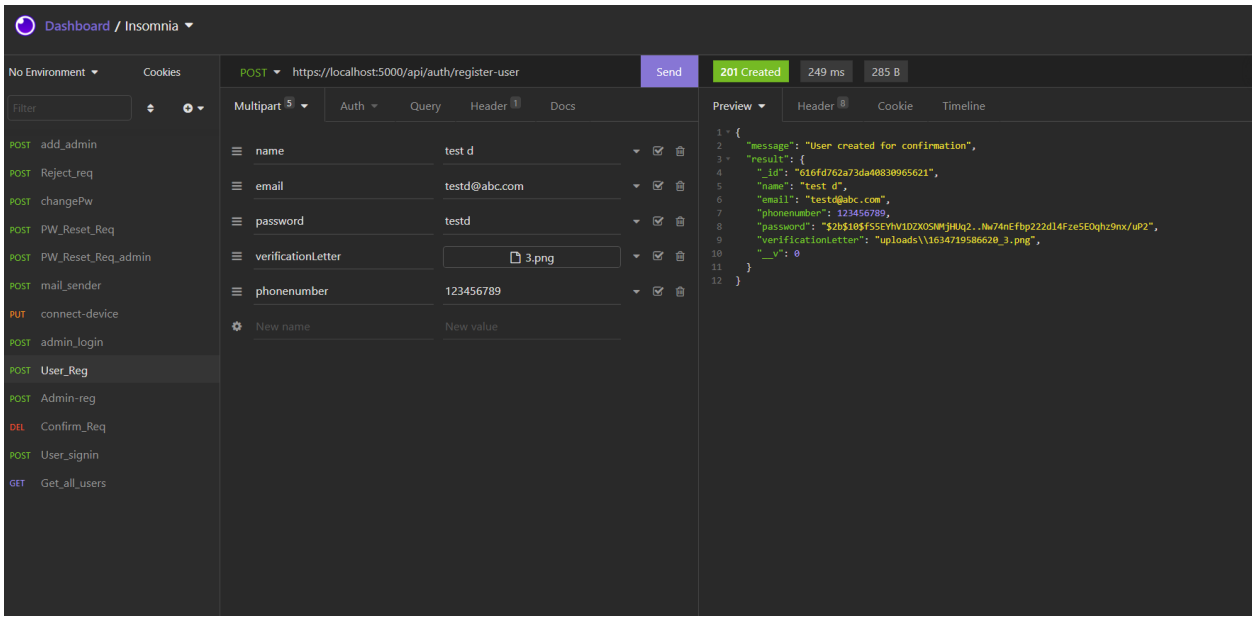
We have used the automated testing tool selenium to test the front end. The first test we have done is the navigation test. We wrote a script to click all possible links in our web application and we have checked whether it navigates to the expected page. Form validations are another important aspect of UI. So, we wanted to test whether our app provides correct feedback when the client fills out a form. For this also wrote a selenium script. In this script, we tested all of our forms with invalid data to make sure they are giving correct feedback. Also, we tested the forms with valid data to make sure that the forms are not giving any invalid feedback when the client inputted valid data.

The scripts are attached here. The testing results are listed below.

Back-End Testing

Prior to the integration of the front and back ends, we wanted to make sure that our rest API returns correct responses to the client's requests. We use 'Insomnia' to test the API. We have implemented a lot of routes in our API to do various tasks and handle various requests from the front end of our web application. It is very important to make sure that the API is providing correct responses. Because the security of the web application depends on these responses. Using insomnia we have tested each route in our API for valid and invalid data.

We have tested whether the API responds with correct status codes, data, and error messages for different data sets. Also as a security test, we passed the wrong credentials to the API using insomnia and checked whether it rejects those requests. We have used JWT tokens to authorize clients. It is very important to make sure that API is rejecting invalid tokens. Therefore we have tested whether the API is rejecting invalid tokens by sending invalid tokens using insomnia. All of these tests are conducted manually. The below figure contains all the cases we have tested using insomnia.



Test Case	Returned Status Code	Expected Status Code	Observation	Faild / Passed
User and Admin login for correct credentials	200	200	Backend responds with status code 200 and send a token to client	passed
User and Admin login for incorrect credentials	401	401	Responds with status code 401 and rejects the request. Just recived credentials invalid message.	passed
User and Admin Registration for emails that are already registered in system	409	409	Rejects registration for same emails.	passed
Password recovery for emails that are not registered in system	401	401	Rejects password recovery for invalid emails.	passed
Password recovery for emails that are registered in system	201	201	Recived message indicating that the password recovery email is sent to entered email.	passed
Try to reset password for second time using a link used before.	401	401	Recived message indicating that the password recovery link is invalid or expiered.	passed
Try to reset password of a accont using a token sent to another email	401	401	Recived message indicating that the password recovery link is invalid or expiered.	passed
Try to register as an admin using a link that already used to register.	401	401	Recived message indicating that the link is invalid or expiered.	passed
Try to log in to the admin dashboard using a token recived for user login	401	401	Recived message indicating that the client is not an admin. Redirection rejected.	passed
Confirm or Reject request with admin token.	200	200	With admin token. Possible to confirm or reject client request.	passed
Confirm or Reject client request with user token.	401	401	With user tokens. Cannot confirm or reject requests. Recieved message unauthorized.	passed

Integration Testing

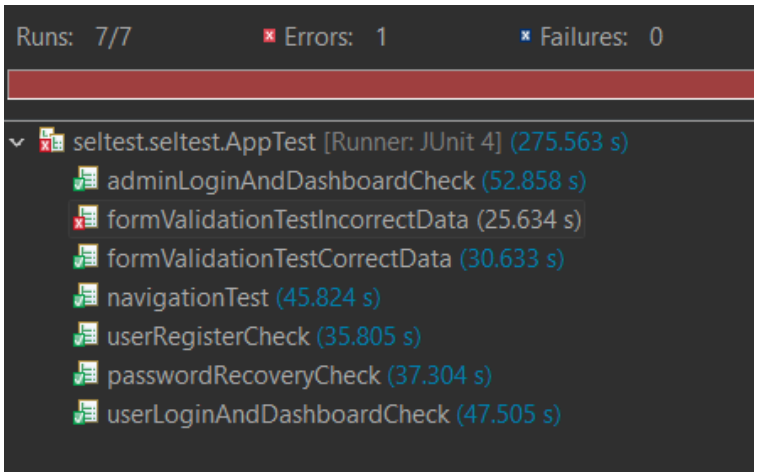
To test the integration of the backend, frontend, and database, we used selenium. We have tested the response from the server to logins, user registrations, password recoveries, and invalid inputs to make sure that the data flow from the front end to the back end and from the back end to the database is fine. We have written scripts to enter valid and invalid credentials in login pages and check the response of the server.

If these component integrations are fine, the logins for valid credentials must be successful. We have tested whether the request with the correct credentials directs the client to the dashboards in the scripts. We had to check the database manually to make sure all the data entered by the script is recorded in the database correctly.

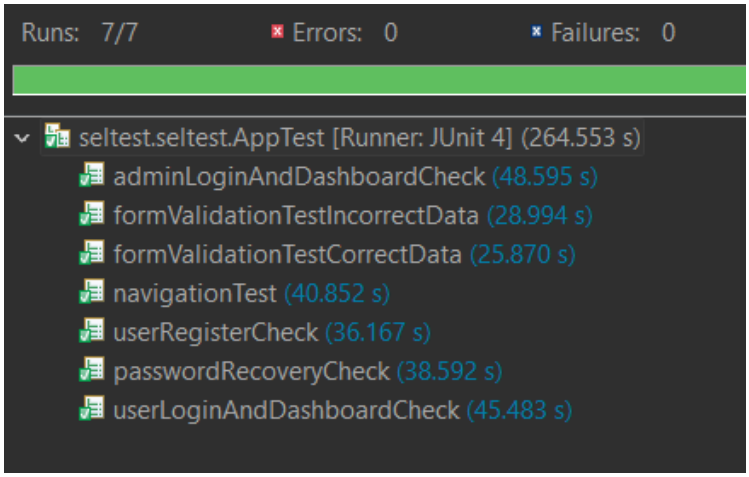
Also, we have tested the same script for three popular browsers. Google Chrome, Microsoft Edge, and Operamini to make sure our web application is compatible with different platforms.

The testing scripts are attached here.

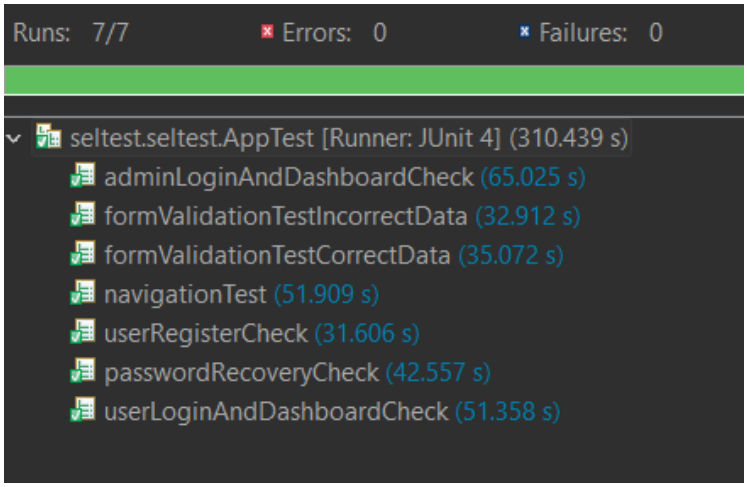
1. Test run No 1: Chrome Browser



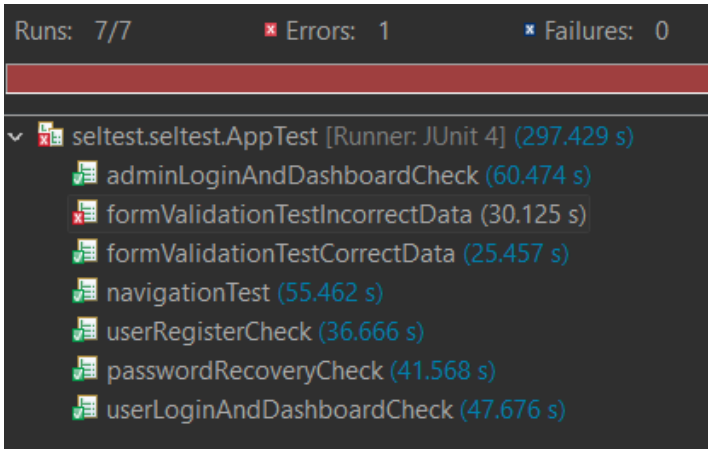
2. Test run No 2: Chrome Browser



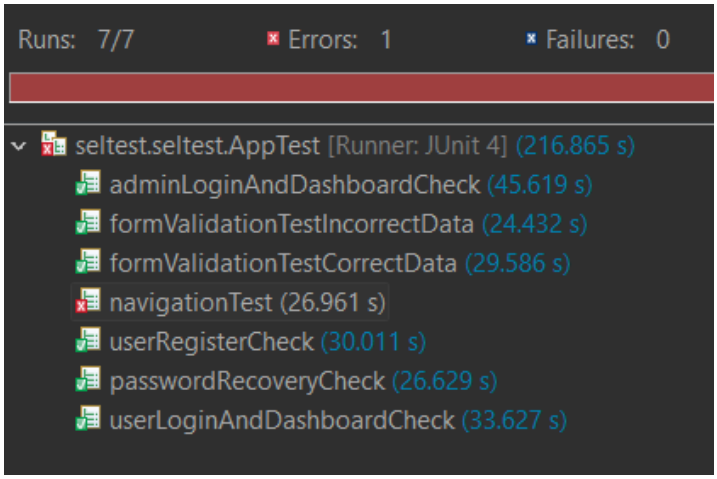
3. Test run No 3: Microsoft Edge Browser



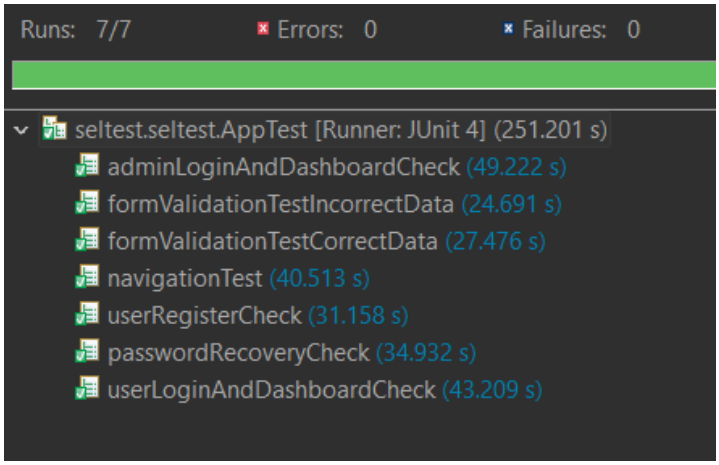
4. Test run No 4: Microsoft Edge Browser



5. Test run No 5: Operamini Browser



6. Test run No 6: Operamini Browser



Test Case	Chrome	Edge	Opera
Navigations to all the pages	passed	passed	passed:1 Faild:1
Form validations for incorrect inputs	passed:1 Faild:1	passed:1 Faild:1	passed
Form validations for correct inputs	passed	passed	passed
User login and dashboard function	passed	passed	passed
User registration test for valid and invalid data	passed	passed	passed
Admin login and dashboard function test	passed	passed	passed
Password recovery test for valid and invalid emails.	passed	passed	passed

Final Product