

Project Report

Project Title	Application Integration – API
Module Name	Application Integration (API using Spring Boot & React JS)
Course Name	Applied Degree in Software Engineering
Module Name (NICF)	Application Integration (API using Spring Boot & React JS)

Student name		Assessor name
Chathushi Jayarathna		MS.Aravinder Kaur
Date issued	Completion date	Submitted on
2 January 2023	11 January 2023	30 January 2023

Project title	Application Integration - API
----------------------	--------------------------------------

Learner declaration
<p>I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.</p> <p>Student signature: Chathushi Date: 30 January 2023</p>

Table of Contents

1) Project Background	5
2) Project Objectives	6
2.1 Project Requirement Specification	6
2.2 Tools & platforms used	7
3) Explain what API is, its role and need for API and research existing APIs.	11
3.1 What is API?	11
3.2 API role and need for API	11
3.3 Research existing APIs	12
4) Relationship between API and SDK.	12
5) Types of API and its uses.	14
6) Identify the potential security issues with API and critically evaluate the suitable API for a given scenario or your selected application.	15
7) Analyze the given scenario, identify the requirements and select the appropriate API.	17
7.1 Requirements in the "Know-Your-Neighborhood" project.	17
8) Develop relevant wireframes for using the API for specific purposes.	18
9) Identify your scope and target platform.	18
10) Evaluate and justify your choice of APIs for your application. (Shows security for the selected API.)	19
10.1 Security of selected API for the application	20
11) Different types of the backend, frontend, and API implementation process	21
12) Discuss a range of suitable development environments for front-end and back-end to develop an application	23
13) Develop a backend and Web service using selected development environment for given scenario	24
15) Construct the application which implements the selected API in Task 2.	35
16) Implement white Box testing for the developed API of your Application	39
16.1 What is white box testing?	39
16.2 Why we use white box testing?	39
17) Conduct Black Box testing (UAT testing) of your developed application and show the evidence for each test case.	41
17.1 What is black box testing?	41
17.2 Benefits of black box testing	42

18)	Review your developed API, and identify the strength and weaknesses of API.	49
19)	Provide data security report of your developed application.....	50
20)	Developed Pages	52

Table of Figures

Figure 1:Screenshot of Spring Boot	7
Figure 2:Screenshot of MySQL Workbench	7
Figure 3:Screenshot of Spring Initializer	8
Figure 4:Screenshot of VS Code	8
Figure 5:Screenshot of Diagram.net	9
Figure 6:Screenshot of Postman	9
Figure 7:Screenshot of MS Word	10
Figure 8:Screenshot of MS PowerPoint	10
Figure 9:Wireframe of Facebook Login.....	18
Figure 10:Server Side Authentication	20
Figure 11:Screenshot of UserPrinciple.java	24
Figure 12:Screenshot of UserDetailsServiceImpl.java	25
Figure 13:Screenshot of AuthController.java	27
Figure 14:Screenshot of FacebookOAuth2UserInfo.java	28
Figure 15:Screenshot of HttpCookieOAuth2AuthorizationRequestRepository.java	29
Figure 16:Screenshot of Auth-context.js	31
Figure 17:Screenshot of FacebookLogin.jsx.....	32
Figure 18:Screenshot of LoginPage.jsx	34
Figure 19: Application Construction using Facebook Login API step 1	35
Figure 20:Application Construction using Facebook Login API step 2	36
Figure 21:Application Construction using Facebook Login API step 3	36
Figure 22:Application Construction using Facebook Login API step 4.....	37
Figure 23:Application Construction using Facebook Login API step 5.....	37
Figure 24: Application Construction using Facebook Login API (Application.yml)	38
Figure 25: Postman Testing - Registration	40
Figure 26:Postman Testing - Login.....	40
Figure 27:Postman Testing - Get user Login	40
Figure 28:Black box testing	41
Figure 29:Screenshot of Home Page.....	52
Figure 30:Screenshot of Login Page	52
Figure 31:Screenshot of Facebook login	53
Figure 32:Screenshot of Registration.....	53
Figure 33:Screenshot of Stores	54
Figure 34:Screenshot of Screenshot Store Details.....	54
Figure 35:Screenshot of Add Stores.....	55
Figure 36:Screenshot of Edit Store	55
Figure 37:Screenshot of Profile Page	56
Figure 38:Screenshot of Edit Profile Page.....	56
Figure 39:Screenshot of About Us Page	57
Figure 40:Screenshot of Contact Us Page.....	57

1) Project Background

The Know-Your-Neighborhood application developed using Spring Boot is enhanced by API access. Rebuilding the application using React.Js and Spring Boot as a grant, demonstrates using APIs to log in and log out. For that purpose, this application should set up a login button with an API. Here Spring Boot framework is used for the backend and react Js is used for the front end. Also the scope here is to test and compare different existing APIs sample wise, assess their suitability, identify potential security issues and use selected APIs on existing websites create a login.

2) Project Objectives

- Be able to design and develop a backend using Spring Boot and JPA Framework.
- Be able to develop API using Restful Web Services.
- Be able to develop frontend applications using React JS.
- Be able to identify existing APIs and their uses in the already-developed application

You have already developed a "Know-Your-Neighborhood" application. The goal of this application is to provide login/sign-up using existing API. For this to happen, the application should have a login button with available APIs

2.1 Project Requirement Specification

The application can use existing APIs to log in and retrieve basic information such as name and email from the API. Users can also register/login manually or via social login (Google / Facebook).

The Know Your Neighborhood website consists of the following main pages:

- Home Page
- Registration Page
- Login Pages with API link
- Contact us Page
- About us Page
- Dashboard

✓ Spring Boot - Backend

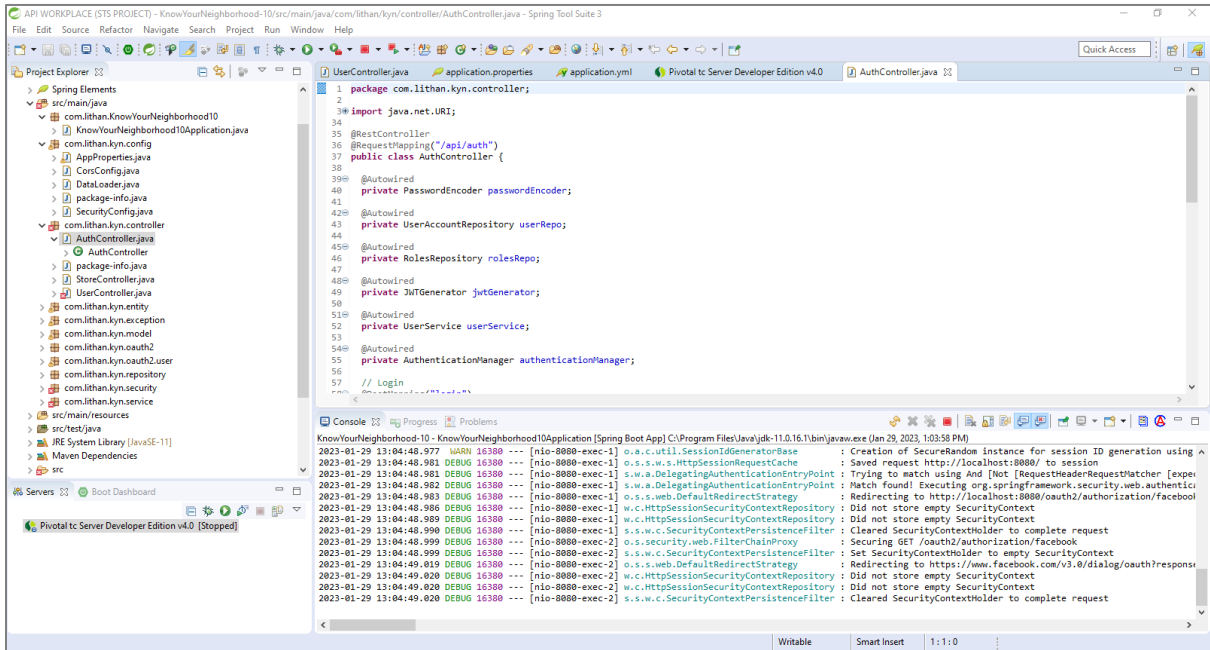


Figure 1: Screenshot of Spring Boot

✓ MySQL Workbench - Database

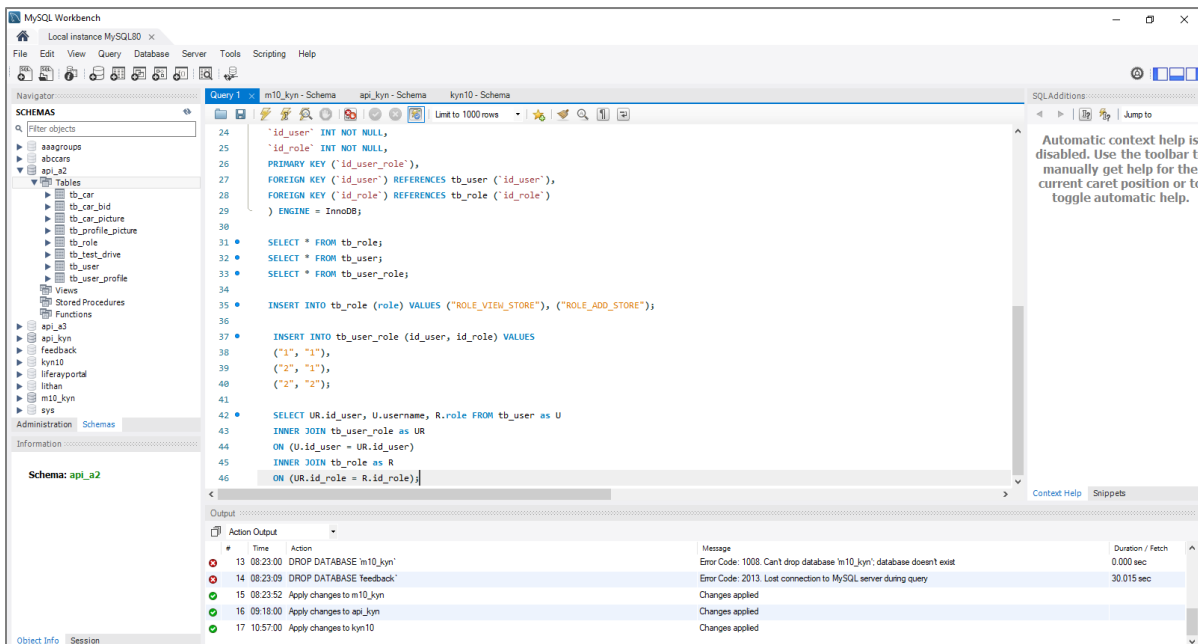


Figure 2: Screenshot of MySQL Workbench

✓ Spring Initializer

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ **Java** ☐ Kotlin ☐ Groovy

☒ **Maven**

Spring Boot

☐ 3.0.3 (SNAPSHOT) ☐ 3.0.2 ☐ 2.7.9 (SNAPSHOT) ☒ **2.7.8**

Project Metadata

Group:

Artifact:

Name:

Description:

Package name:

Packaging: ☒ **Jar** ☐ War

Java: ☐ 19 ☐ 17 ☒ **11** ☐ 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Spring Security SECURITY
Highly customizable authentication and access-control framework for Spring applications.

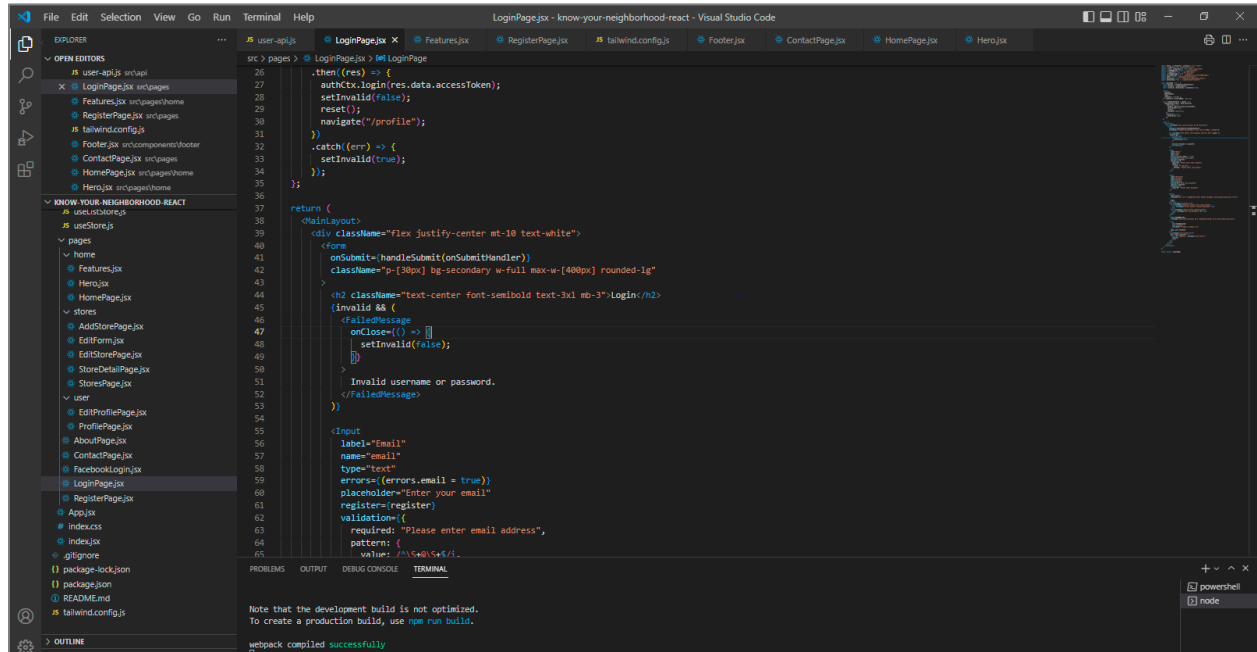
MySQL Driver SQL
MySQL JDBC driver.

Spring Boot Actuator OPS
Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

Buttons: GENERATE CTRL + G, EXPLORE CTRL + SPACE, SHARE...

Figure 3: Screenshot of Spring Initializer

✓ Visual Studio Code - Frontend



✓ Diagram.Net

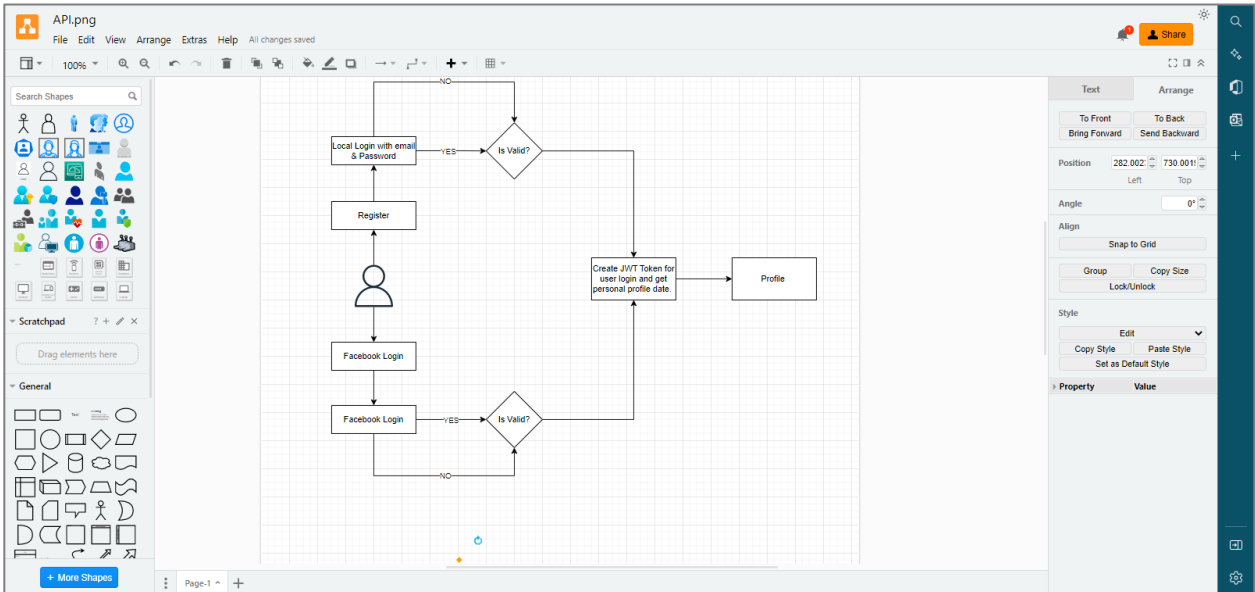


Figure 5: Screenshot of Diagram.net

✓ Postman

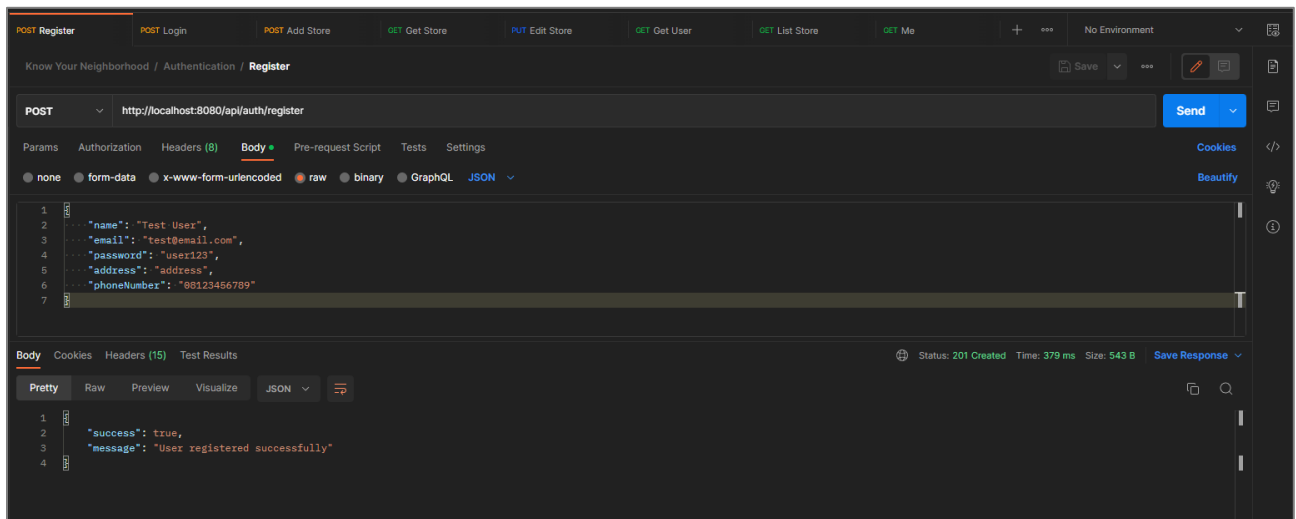


Figure 6: Screenshot of Postman

✓ Microsoft Word

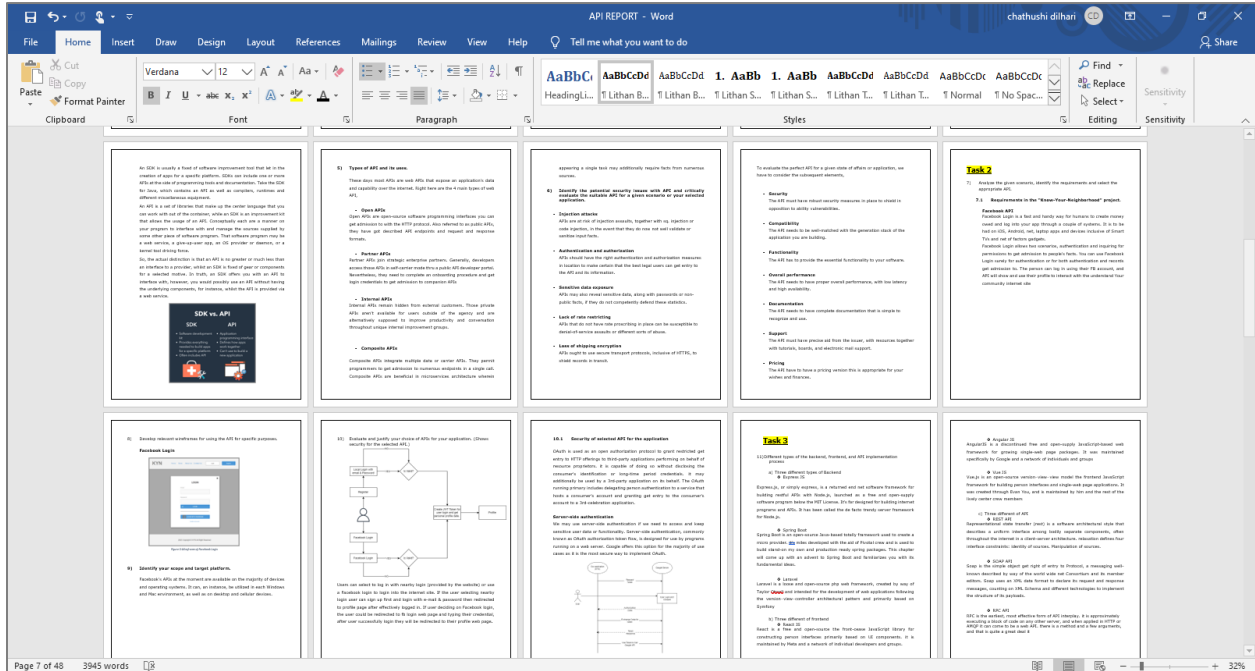


Figure 7: Screenshot of MS Word

✓ Microsoft PowerPoint

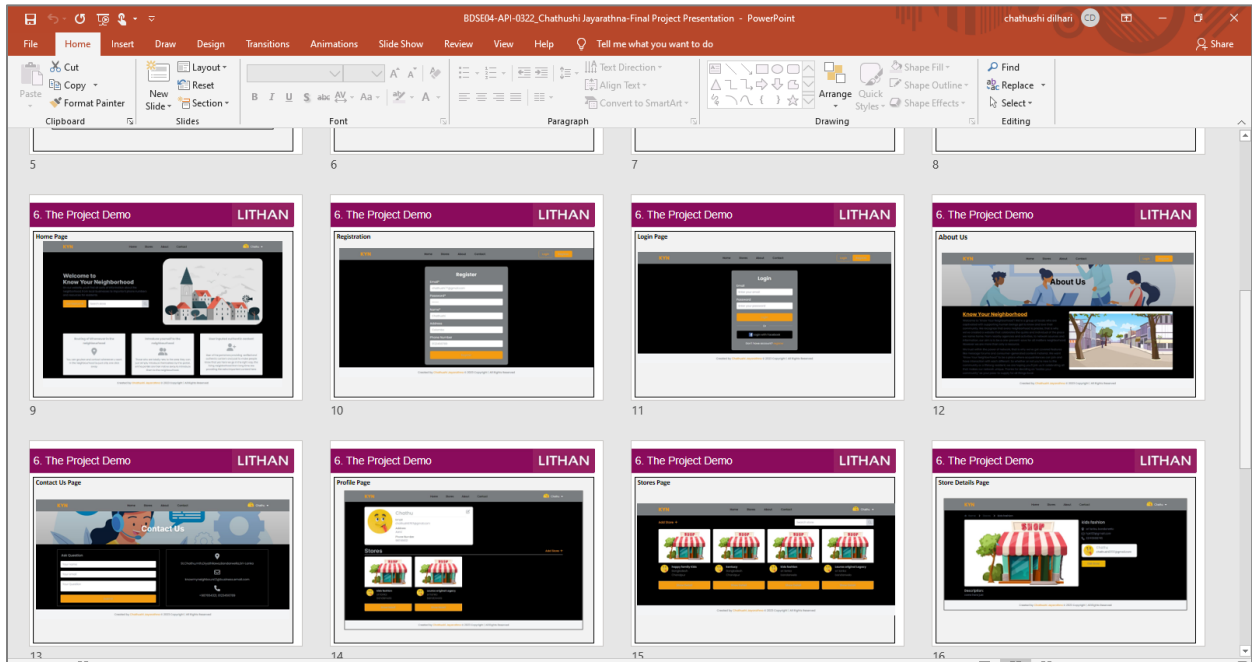


Figure 8: Screenshot of MS PowerPoint

Task 1

- 3) Explain what API is, its role and need for API and research existing APIs.

3.1 What is API?

An application programming interface (API) is code that permits two software program packages to communicate. An API defines how a developer should request offerings from an operating machine (OS) or different utility, and expose data within one-of-a-kind contexts and throughout more than one channels.

Any statistics can be shared with an utility programming interface. APIs are implemented via function calls composed of verbs and nouns; the specified syntax is described in the documentation of the software being called. for example, on a actual estate internet site, one API would possibly put up available actual estate houses via geography, while a 2nd API provides current hobby quotes and a third offers a mortgage calculator.

3.2 API role and need for API

APIs are used in a huge variety of contexts, such as operating systems, application software programs, and web-based systems. They may be used to permit distinct software program packages to communicate with every different, to permit a front-end web application to access information from a back-end database or to allow a cell app to connect to a community-based service.

The want for APIs arises because extraordinary software systems regularly want to be incorporated with each other to be able to change statistics or provide positive services. as an example, a cellular app that enables users to discover nearby restaurants may use an API to fetch data from a database of restaurant listings. without an API, the mobile app would must immediately get entry to the database, which would be difficult to put into effect and maintain.

3.3 Research existing APIs

There are many exclusive APIs available for an extensive range of functions. A few APIs are open and to be had to anyone, at the same time as others are proprietary and only available to unique businesses or people. To analyze current APIs, you could seek online directories inclusive of programmable web or Rapid API, which list thousands of APIs in various classes. You may also look for APIs particular to a specific carrier or generation, including the Google Maps API or the Facebook API.

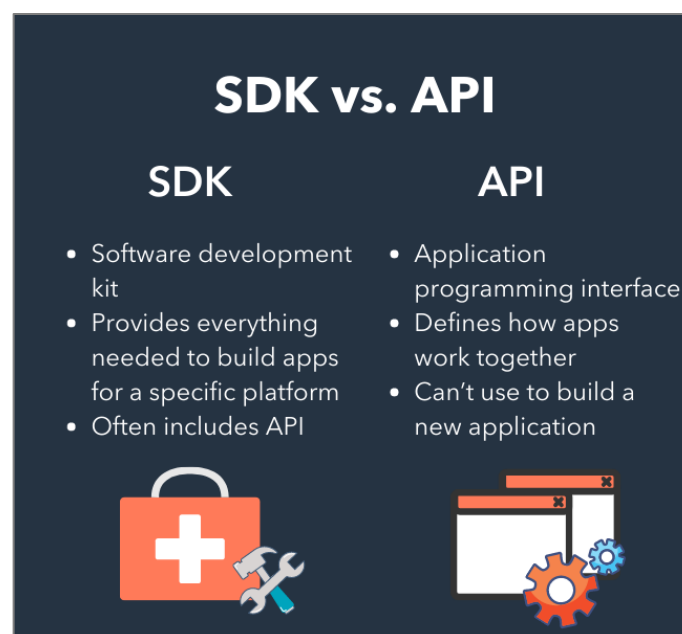
4) Relationship between API and SDK.

Two terms in software development that are frequently confused are API and SDK. API stands for “application Programming Interface” and refers to programming commands and standards for accessing a web tool or database. For example, a software program enterprise will often release its API publicly or privately to different software builders in order to design products powered by its service. An API may be packaged in an SDK, or “software improvement package “.

An SDK is usually a fixed of software improvement tool that let in the creation of apps for a specific platform. SDKs can include one or more APIs at the side of programming tools and documentation. Take the SDK for Java, which contains an API as well as compilers, runtimes and different miscellaneous equipment.

An API is a set of libraries that make up the center language that you can work with out of the container, while an SDK is an improvement kit that allows the usage of an API. Conceptually each are a manner on your program to interface with and manage the sources supplied by some other piece of software program. That software program may be a web service, a give-up-user app, an OS provider or daemon, or a kernel tool driving force.

So, the actual distinction is that an API is no greater or much less than an interface to a provider, whilst an SDK is fixed of gear or components for a selected motive. In truth, an SDK offers you with an API to interface with, however, you would possibly use an API without having the underlying components, for instance, whilst the API is provided via a web service.



5) **Types of API and its uses.**

These days most APIs are web APIs that expose an application's data and capability over the internet. Right here are the 4 main types of web API,

- **Open APIs**

Open APIs are open-source software programming interfaces you can get admission to with the HTTP protocol. Also referred to as public APIs, they have got described API endpoints and request and response formats.

- **Partner APIs**

Partner APIs join strategic enterprise partners. Generally, developers access those APIs in self-carrier mode thru a public API developer portal. Nevertheless, they need to complete an onboarding procedure and get login credentials to get admission to companion APIs

- **Internal APIs**

Internal APIs remain hidden from external customers. Those private APIs aren't available for users outside of the agency and are alternatively supposed to improve productivity and conversation throughout unique internal improvement groups.

- **Composite APIs**

Composite APIs integrate multiple data or carrier APIs. They permit programmers to get admission to numerous endpoints in a single call. Composite APIs are beneficial in microservices architecture wherein

appearing a single task may additionally require facts from numerous sources.

6) Identify the potential security issues with API and critically evaluate the suitable API for a given scenario or your selected application.

- **Injection attacks**

APIs are at risk of injection assaults, together with sq. injection or code injection, in the event that they do not well validate or sanitize input facts.

- **Authentication and authorization**

APIs should have the right authentication and authorization measures in location to make certain that the best legal users can get entry to the API and its information.

- **Sensitive data exposure**

APIs may also reveal sensitive data, along with passwords or non-public facts, if they do not competently defend these statistics.

- **Lack of rate restricting**

APIs that do not have rate proscribing in place can be susceptible to denial-of-service assaults or different sorts of abuse.

- **Loss of shipping encryption**

APIs ought to use secure transport protocols, inclusive of HTTPS, to shield records in transit.

To evaluate the perfect API for a given state of affairs or application, we have to consider the subsequent elements,

- **Security**

The API must have robust security measures in place to shield in opposition to ability vulnerabilities.

- **Compatibility**

The API needs to be well-matched with the generation stack of the application you are building.

- **Functionality**

The API has to provide the essential functionality to your software.

- **Overall performance**

The API needs to have proper overall performance, with low latency and high availability.

- **Documentation**

The API needs to have complete documentation that is simple to recognize and use.

- **Support**

The API must have precise aid from the issuer, with resources together with tutorials, boards, and electronic mail support.

- **Pricing**

The API have to have a pricing version this is appropriate for your wishes and finances.

Task 2

- 7) Analyze the given scenario, identify the requirements and select the appropriate API.

7.1 Requirements in the “Know-Your-Neighborhood” project.

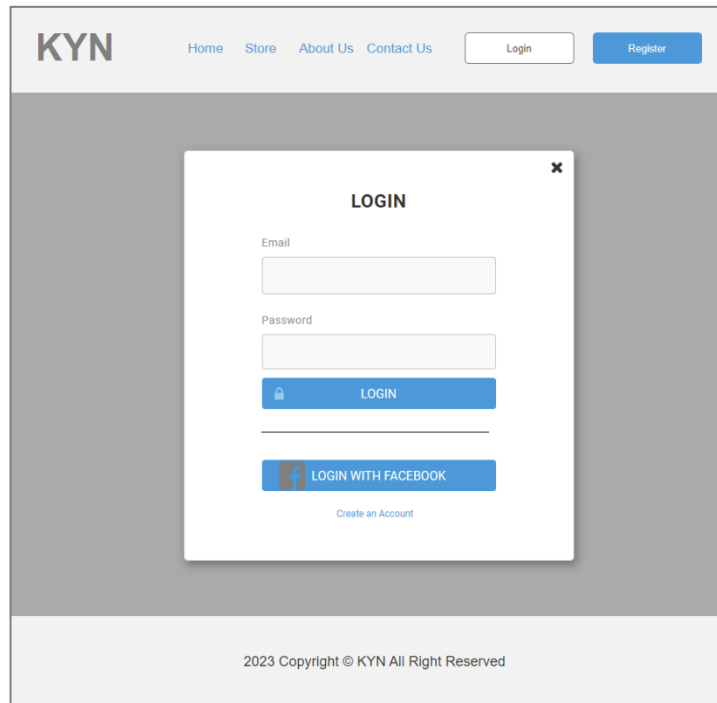
Facebook API

Facebook Login is a fast and handy way for humans to create money owed and log into your app through a couple of systems. It is to be had on iOS, Android, net, laptop apps and devices inclusive of Smart TVs and net of factors gadgets.

Facebook Login allows two scenarios, authentication and inquiring for permissions to get admission to people's facts. You can use Facebook Login surely for authentication or for both authentication and records get admission to. The person can log in using their FB account, and API will show and use their profile to interact with the understand Your community internet site

- 8) Develop relevant wireframes for using the API for specific purposes.

Facebook Login



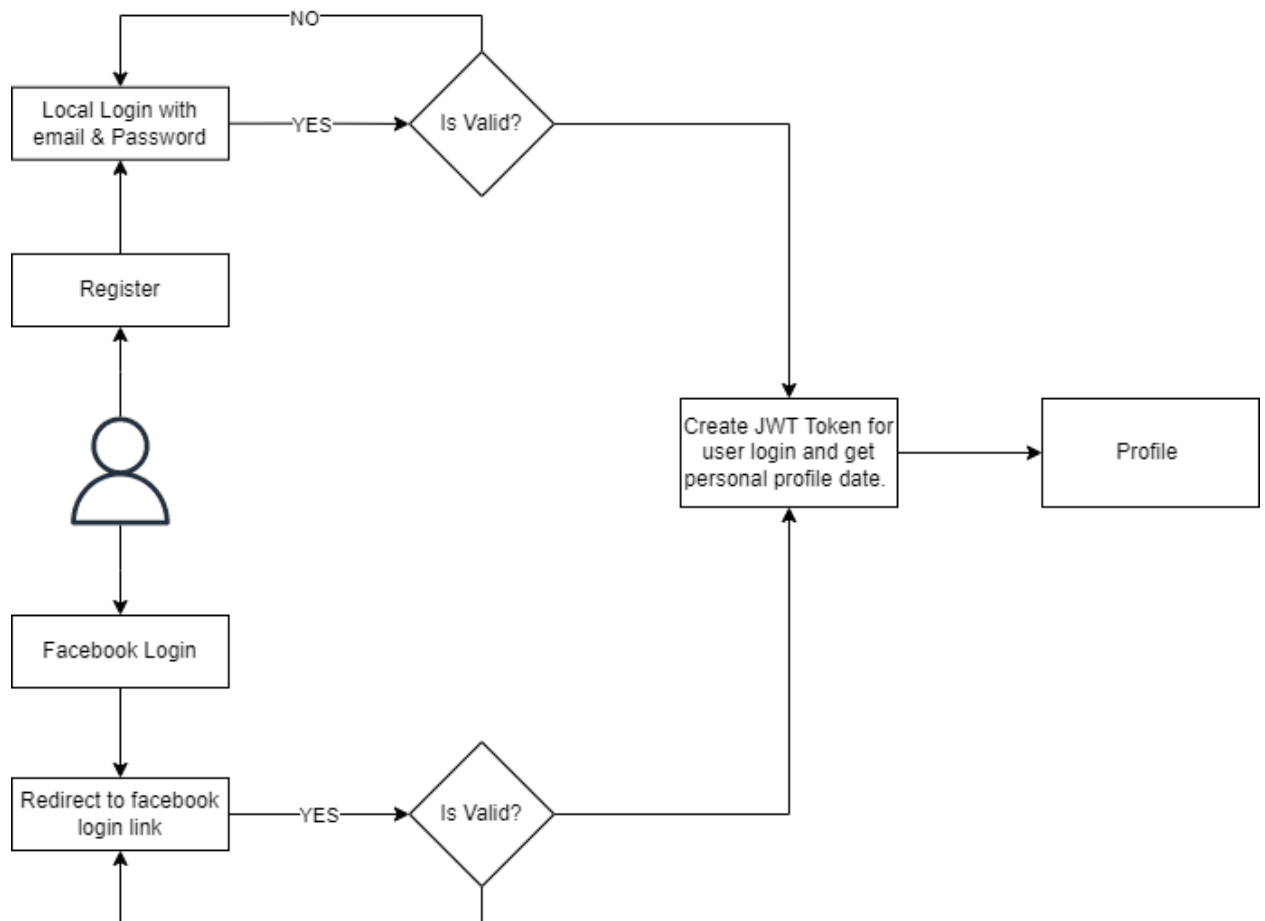
The wireframe shows a web page for 'KYN' with a navigation bar containing 'Home', 'Store', 'About Us', 'Contact Us', 'Login', and 'Register'. A central modal window titled 'LOGIN' contains an 'Email' input field, a 'Password' input field, a 'LOGIN' button with a lock icon, a 'LOGIN WITH FACEBOOK' button with the Facebook logo, and a 'Create an Account' link. The footer of the page reads '2023 Copyright © KYN All Right Reserved'.

Figure 9: Wireframe of Facebook Login

- 9) **Identify your scope and target platform.**

Facebook's APIs at the moment are available on the majority of devices and operating systems. It can, an instance, be utilized in each Windows and Mac environment, as well as on desktop and cellular devices.

- 10) Evaluate and justify your choice of APIs for your application. (Shows security for the selected API.)



Users can select to log in with nearby login (provided by the website) or use a Facebook login to login into the internet site. If the user selecting nearby login user can sign up first and login with e-mail & password then redirected to profile page after effectively logged in. If user deciding on Facebook login, the user could be redirected to fb login web page and typing their credential, after user successfully login they will be redirected to their profile web page.

10.1 Security of selected API for the application

OAuth is used as an open authorization protocol to grant restricted get entry to HTTP offerings to third-party applications performing on behalf of resource proprietors. it is capable of doing so without disclosing the consumer's identification or long-time period credentials. it may additionally be used by a 3rd-party application on its behalf. The OAuth running primary includes delegating person authentication to a service that hosts a consumer's account and granting get entry to the consumer's account to a 3rd-celebration application.

Server-side authentication

We may use server-side authentication if we need to access and keep sensitive user data or functionality. Server-side authentication, commonly known as OAuth authorization token flow, is designed for use by programs running on a web server. Google offers this option for the majority of use cases as it is the most secure way to implement OAuth.

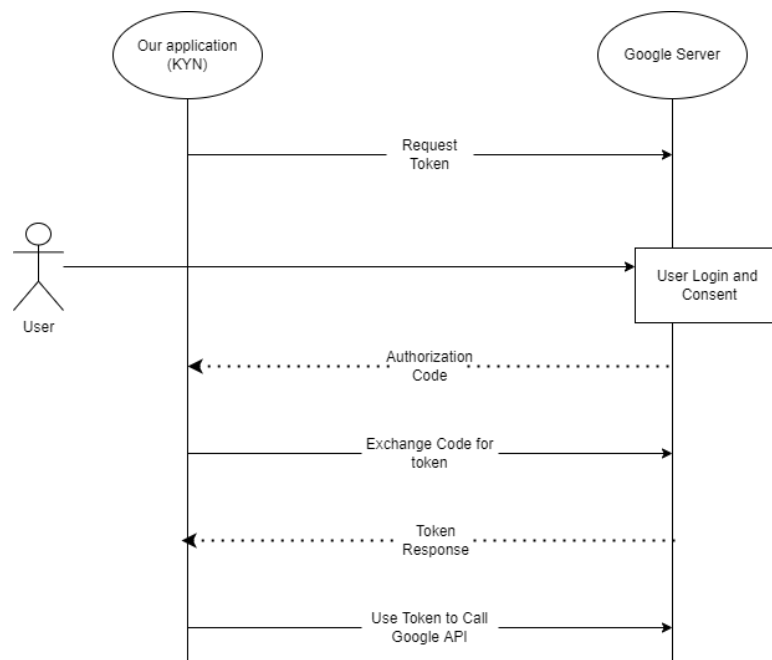


Figure 10:Server Side Authentication

Task 3

11) Different types of the backend, frontend, and API implementation process

a) Three different types of Backend

❖ Express JS

Express.js, or simply express, is a returned end net software framework for building restful APIs with Node.js, launched as a free and open-supply software program below the MIT License. It's far designed for building internet programs and APIs. It has been called the de facto trendy server framework for Node.js.

❖ Spring Boot

Spring Boot is an open-source Java-based totally framework used to create a micro provider. it's miles developed with the aid of Pivotal crew and is used to build stand-on my own and production ready spring packages. This chapter will come up with an advent to Spring Boot and familiarizes you with its fundamental ideas.

❖ Laravel

Laravel is a loose and open-source php web framework, created by way of Taylor Otwell and intended for the development of web applications following the version-view-controller architectural pattern and primarily based on Symfony

b) Three different of frontend

❖ React JS

React is a free and open-source the front-cease JavaScript library for constructing person interfaces primarily based on UI components. it is maintained by Meta and a network of individual developers and groups.

❖ Angular JS

AngularJS is a discontinued free and open-supply JavaScript-based web framework for growing single-web page packages. It was maintained specifically by Google and a network of individuals and groups

❖ Vue JS

Vue.js is an open-source version-view-view model the frontend JavaScript framework for building person interfaces and single-web page applications. It was created through Evan You, and is maintained by him and the rest of the lively center crew members

c) Three different of API

❖ REST API

Representational state transfer (rest) is a software architectural style that describes a uniform interface among bodily separate components, often throughout the internet in a client-server architecture. relaxation defines four interface constraints: identity of sources. Manipulation of sources.

❖ SOAP API

Soap is the simple object get right of entry to Protocol, a messaging well-known described by way of the world wide net Consortium and its member editors. Soap uses an XML data format to declare its request and response messages, counting on XML Schema and different technologies to implement the structure of its payloads.

❖ RPC API

RPC is the earliest, most effective form of API interplay. it is approximately executing a block of code on any other server, and when applied in HTTP or AMQP it can come to be a web API. there is a method and a few arguments, and that is quite a great deal it

12) Discuss a range of suitable development environments for front-end and back-end to develop an application

a) Front-end Development

❖ Visual Studio Code

Visual Studio Code combines the simplicity of a source code editor with effective developer tooling, like intelligence code of completion and debugging. First and major, it's far an editor that receives from your way. The delightfully frictionless edit-construct-debug cycle approach less time fiddling with your surroundings, and extra time executing in your thoughts

❖ WebStorm

WebStorm is an incorporated development environment for JavaScript and related technologies. Like different JetBrains IDEs, it makes your development experience extra exciting, automating routine work and assisting you manage complex tasks effortlessly.

b) Backend Development

❖ Visual Studio

A general-purpose development environment that can be used for back-end development with a variety of languages, including C# and .NET.

❖ Spring Tool Suite

Spring tool Suite (STS) is a java IDE tailored for developing Spring-primarily based organization programs. it is less complicated, quicker, and extra convenient. And most importantly it's far based on Eclipse IDE. STS is unfastened, open-supply, and powered with the aid of VMware

13) Develop a backend and Web service using selected development environment for given scenario

UserPrincipal.java

```
14
15 public class UserPrincipal implements OAuth2User, UserDetails {
16     private int id;
17     private String email;
18     private String password;
19     private Collection<? extends GrantedAuthority> authorities;
20     private Map<String, Object> attributes;
21
22     public UserPrincipal(int id, String email, String password, Collection<? extends
GrantedAuthority> authorities) {
23         this.id = id;
24         this.email = email;
25         this.password = password;
26         this.authorities = authorities;
27     }
28
29     public static UserPrincipal create(UserAccount user) {
30         List<GrantedAuthority> authorities = user.getRoles().stream()
31             .map((role -> new
SimpleGrantedAuthority(role.getName().name()))).collect(Collectors.toList());
32
33         return new UserPrincipal(
34             user.getUserId(),
35             user.getEmail(),
36             user.getPassword(),
37             authorities);
38     }
39
40     public static UserPrincipal create(UserAccount user, Map<String, Object>
attributes) {
41         UserPrincipal userPrincipal = UserPrincipal.create(user);
42         userPrincipal.setAttributes(attributes);
43         return userPrincipal;
44     }
45
46     public int getId() {
47         return id;
48     }
49
50     public String getEmail() {
51         return email;
52     }
53
54     @Override
55     public String getPassword() {
56         return password;
57     }
58
59     @Override
60     public String getUsername() {
61         return email;
62     }
63
64     @Override
65     public boolean isAccountNonExpired() {
66         return true;
67     }
68
69     @Override
70     public boolean isAccountNonLocked() {
71         return true;
72     }
73
74     @Override
75     public boolean isCredentialsNonExpired() {
76         return true;
77     }
78
79     @Override
80     public boolean isEnabled() {
81         return true;
82     }
83
84     @Override
85     public Collection<? extends GrantedAuthority> getAuthorities() {
86         return authorities;
87     }
88
89     @Override
90     public Map<String, Object> getAttributes() {
91         return attributes;
92     }
93
94     public void setAttributes(Map<String, Object> attributes) {
95         this.attributes = attributes;
96     }
97
98     @Override
99     public String getName() {
100         return String.valueOf(id);
101     }
102 }
```

Figure 11: Screenshot of UserPrincipal.java

UserDetailsServiceImpl.java

```
package com.lithan.kyn.security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import com.lithan.kyn.entity.UserAccount;
import com.lithan.kyn.exception.ResourceNotFoundException;
import com.lithan.kyn.repository.UserAccountRepository;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserAccountRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String email) throws
    UsernameNotFoundException {

        UserAccount user = userRepository.findByEmail(email)
            .orElseThrow(() -> new UsernameNotFoundException("User not found with email :
" + email));

        return UserPrincipal.create(user);
    }

    public UserDetails loadUserById(int id) {
        UserAccount user = userRepository.findById(id).orElseThrow(
            () -> new ResourceNotFoundException("User", "id", id));

        return UserPrincipal.create(user);
    }
}
```

Figure 12: Screenshot of UserDetailsServiceImpl.java

AuthController.java

```
package com.lithan.kyn.controller;

import java.net.URI;
import java.util.Arrays;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import com.lithan.kyn.entity.EAuthProvider;
import com.lithan.kyn.entity.ERole;
import com.lithan.kyn.entity.Roles;
import com.lithan.kyn.entity.UserAccount;
import com.lithan.kyn.model.ApiResponse;
import com.lithan.kyn.model.AuthResponse;
import com.lithan.kyn.model.LoginDto;
import com.lithan.kyn.model.RegisterDto;
import com.lithan.kyn.repository.RolesRepository;
import com.lithan.kyn.repository.UserAccountRepository;
import com.lithan.kyn.security.JWTGenerator;
import com.lithan.kyn.service.UserService;

@RestController
@RequestMapping("/api/auth")
public class AuthController {

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private UserAccountRepository userRepo;

    @Autowired
    private RolesRepository rolesRepo;

    @Autowired
    private JWTGenerator jwtGenerator;

    @Autowired
    private UserService userService;

    @Autowired
    private AuthenticationManager authenticationManager;

    // Login
    @PostMapping("login")
```

```

public ResponseEntity<AuthResponse> login(@Valid @RequestBody LoginDto loginDto) {
    Authentication authentication = authenticationManager.authenticate(
        new UsernamePasswordAuthenticationToken(
            loginDto.getEmail(),
            loginDto.getPassword()));

    SecurityContextHolder.getContext().setAuthentication(authentication);
    String token = jwtGenerator.generateToken(authentication);
    return new ResponseEntity<>(new AuthResponse(token), HttpStatus.OK);
}

// Register
@PostMapping("register")
public ResponseEntity<?> addUser(@Valid @RequestBody RegisterDto registerDto)
throws Exception {

    if (userRepo.existsByEmail(registerDto.getEmail())) {
        return new ResponseEntity<>("Email is taken!", HttpStatus.BAD_REQUEST);
    }

    UserAccount user = new UserAccount();
    user.setName(registerDto.getName());
    user.setEmail(registerDto.getEmail());
    user.setAddress(registerDto.getAddress());
    user.setPhoneNumber(registerDto.getPhoneNumber());
    user.setPassword(passwordEncoder.encode(registerDto.getPassword()));
    user.setProvider(EAuthProvider.local);

    Roles role = rolesRepo.findByName(ERole.ROLE_USER);
    if (role == null) {
        role = userService.createRole(ERole.ROLE_USER);
    }
    user.setRoles(Arrays.asList(role));

    UserAccount newUser = userRepo.save(user);

    URI location = ServletUriComponentsBuilder
        .fromCurrentContextPath().path("api/users/me")
        .buildAndExpand(newUser.getUserId()).toUri();

    return ResponseEntity.created(location)
        .body(new ApiResponse(true, "User registered successfully"));
}
}

```

Figure 13: Screenshot of AuthController.java

Backend Facebook Login

FacebookOAuth2UserInfo.java

```
package com.lithan.kyn.oauth2.user;

import java.util.Map;

public class FacebookOAuth2UserInfo extends OAuth2UserInfo {
    public FacebookOAuth2UserInfo(Map<String, Object> attributes) {
        super(attributes);
    }

    @Override
    public String getId() {
        return (String) attributes.get("id");
    }

    @Override
    public String getName() {
        return (String) attributes.get("name");
    }

    @Override
    public String getEmail() {
        return (String) attributes.get("email");
    }

    @Override
    public String getImageUrl() {
        if (attributes.containsKey("picture")) {
            Map<String, Object> pictureObj = (Map<String, Object>)
attributes.get("picture");
            if (pictureObj.containsKey("data")) {
                Map<String, Object> dataObj = (Map<String, Object>) pictureObj.get("data");
                if (dataObj.containsKey("url")) {
                    return (String) dataObj.get("url");
                }
            }
        }
        return null;
    }
}
```

Figure 14: Screenshot of FacebookOAuth2UserInfo.java

HttpCookieOAuth2AuthorizationRequestRepository.java

```
@Component
public class HttpCookieOAuth2AuthorizationRequestRepository
    implements AuthorizationRequestRepository<OAuth2AuthorizationRequest> {
    public static final String OAUTH2_AUTHORIZATION_REQUEST_COOKIE_NAME =
"oauth2_auth_request";
    public static final String REDIRECT_URI_PARAM_COOKIE_NAME = "redirect_uri";
    private static final int cookieExpireSeconds = 180;

    @Override
    public OAuth2AuthorizationRequest loadAuthorizationRequest(HttpServletRequest
request) {
        return CookieUtils.getCookie(request, OAUTH2_AUTHORIZATION_REQUEST_COOKIE_NAME)
            .map(cookie -> CookieUtils.deserialize(cookie,
OAuth2AuthorizationRequest.class))
            .orElse(null);
    }

    @Override
    public void saveAuthorizationRequest(OAuth2AuthorizationRequest
authorizationRequest, HttpServletRequest request,
        HttpServletResponse response) {
        if (authorizationRequest == null) {
            CookieUtils.deleteCookie(request, response,
OAUTH2_AUTHORIZATION_REQUEST_COOKIE_NAME);
            CookieUtils.deleteCookie(request, response, REDIRECT_URI_PARAM_COOKIE_NAME);
            return;
        }

        CookieUtils.addCookie(response, OAUTH2_AUTHORIZATION_REQUEST_COOKIE_NAME,
            CookieUtils.serialize(authorizationRequest), cookieExpireSeconds);
        String redirectUriAfterLogin =
request.getParameter(REDIRECT_URI_PARAM_COOKIE_NAME);
        if (StringUtils.isNotBlank(redirectUriAfterLogin)) {
            CookieUtils.addCookie(response, REDIRECT_URI_PARAM_COOKIE_NAME,
redirectUriAfterLogin, cookieExpireSeconds);
        }
    }

    @Override
    public OAuth2AuthorizationRequest removeAuthorizationRequest(HttpServletRequest
request) {
        return this.loadAuthorizationRequest(request);
    }

    public void removeAuthorizationRequestCookies(HttpServletRequest request,
        HttpServletResponse response) {
        CookieUtils.deleteCookie(request, response,
OAUTH2_AUTHORIZATION_REQUEST_COOKIE_NAME);
        CookieUtils.deleteCookie(request, response, REDIRECT_URI_PARAM_COOKIE_NAME);
    }
}
```

Figure 15: Screenshot of HttpCookieOAuth2AuthorizationRequestRepository.java

14) Develop an application that utilizes an API.

Developed Application using React

❖ Auth-context.js

```
import React, { useEffect, useState } from "react";
import { getUserLoginAPI } from "../api/user-api";

// Create Context API
const AuthContext = React.createContext({
  userId: "",
  profile: {
    userId: "",
    name: "",
    email: "",
    imageUrl: "",
    address: "",
    phoneNumber: "",
  },
  stores: [
    {
      storeId: "",
      storeName: "",
      country: "",
      city: "",
      storeEmail: "",
      phoneNumber: "",
    },
  ],
  roles: [],
  token: "",
  isLoggedIn: false,
  login: (token) => {},
  logout: () => {},
  refresh: () => {},
});

// Retrive Stored Token From Local Storage
const retrieveStoredToken = () => {
  const storedToken = localStorage.getItem("token");

  return {
    token: storedToken,
  };
};

// Context Provider
export const AuthContextProvider = (props) => {
  const storedToken = retrieveStoredToken();
  let initialToken;

  if (storedToken) {
    initialToken = storedToken.token;
  }

  // USE STATE
  const [token, setToken] = useState(initialToken);
  const [userProfile, setUserProfile] = useState({});
  const [userStores, setUserStores] = useState([]);
  const [userRoles, setUserRoles] = useState([]);

  // Check Token (!! = convert to Boolean)
  const userIsLoggedIn = !!token;
```

```

// GET USER LOGIN DATA
useEffect(() => {
  if (token) {
    getUserLoginAPI(token)
      .then((res) => {
        setUserProfile(res.data.profile);
        setUserStores(res.data.stores);
        setUserRoles(res.data.roles);
      })
      .catch((err) => {
        console.log(err);
        logoutHandler();
      });
  }
}, [token]);

// LOGOUT
const logoutHandler = () => {
  setToken(null);
  localStorage.removeItem("token");
};

// LOGIN
const loginHandler = (token) => {
  setToken(token);
  localStorage.setItem("token", token);
};

// REFRESH USER LOGIN DATA
const refreshHandler = () => {
  getUserLoginAPI(token)
    .then((res) => {
      setUserProfile(res.data.profile);
      setUserStores(res.data.stores);
      setUserRoles(res.data.roles);
    })
    .catch((err) => {
      console.log(err);
      logoutHandler();
    });
};

// Context Value
const contextValue = {
  userId: userProfile.userId,
  profile: userProfile,
  stores: userStores,
  roles: userRoles,
  token: token,
  isLoggedIn: userIsLoggedIn,
  login: loginHandler,
  logout: logoutHandler,
  refresh: refreshHandler,
};

return (
  <AuthContext.Provider value={contextValue}>
    {props.children}
  </AuthContext.Provider>
);

```

Figure 16: Screenshot of Auth-context.js

❖ FacebookLogin.jsx

```
import React, { useContext, useEffect } from "react";
import { useNavigate, useSearchParams } from "react-router-dom";
import AuthContext from "../context/auth-context";

const FacebookLogin = () => {
  const authCtx = useContext(AuthContext);
  const [searchParams] = useSearchParams();
  const navigate = useNavigate();

  useEffect(() => {
    if (searchParams.get("token") !== null) {
      authCtx.login(searchParams.get("token"));
      navigate("/profile");
    }
  }, [authCtx, navigate, searchParams]);

  return <div className="mt-10 text-center">Loading...</div>;
};

export default FacebookLogin;
```

Figure 17: Screenshot of FacebookLogin.jsx

LoginPage.jsx

```
import React, { useContext, useState } from "react";
import { useForm } from "react-hook-form";
import { Link, useNavigate } from "react-router-dom";
import { FACEBOOK_URL } from "../api/constant";
import { loginAPI } from "../api/user-api";
import { facebookLogo } from "../assets";
import FailedMessage from "../components/form/FailedMessage";
import Input from "../components/form/Input";
import MainLayout from "../components/layout/MainLayout";
import AuthContext from "../context/auth-context";

const LoginPage = () => {
  const authCtx = useContext(AuthContext);
  const navigate = useNavigate();
  const [invalid, setInvalid] = useState(false);

  const {
    register,
    handleSubmit,
    reset,
    formState: { errors },
  } = useForm({ criteriaMode: "all" });

  const onSubmitHandler = (data) => {
    loginAPI(data.email, data.password)
      .then((res) => {
        authCtx.login(res.data.accessToken);
        setInvalid(false);
        reset();
        navigate("/profile");
      })
      .catch((err) => {
        setInvalid(true);
      });
  };

  return (
    <MainLayout>
      <div className="flex justify-center mt-10 text-white">
        <form
          onSubmit={handleSubmit(onSubmitHandler)}
          className="p-[30px] bg-secondary w-full max-w-[400px] rounded-lg"
        >
          <h2 className="text-center font-semibold text-3xl mb-3">Login</h2>
          {invalid && (
            <FailedMessage
              onClose={() => {
                setInvalid(false);
              }}
            >
              Invalid username or password.
            </FailedMessage>
          )}

          <Input
            label="Email"
            name="email"
            type="text"
            errors={errors.email ? true : false}
          />
        </form>
      </div>
    </MainLayout>
  );
};
```

```

        placeholder="Enter your email"
        register={register}
        validation={{
          required: "Please enter email address",
          pattern: {
            value: /^[S+@\S+$/i,
            message: "Please enter valid email",
          },
        }}
      />

      <Input
        label="Password"
        name="password"
        type="password"
        errors={errors}
        placeholder="Enter your password"
        register={register}
        validation={{
          required: "Please enter password",
        }}
      />

      <button
        type="submit"
        className="py-3 mt-3 rounded-md border border-secondary text-primary bg-
color1 w-full"
      >
        Login
      </button>
      <div className="relative mt-4">
        <div className="absolute inset-0 flex items-center">
          <div className="w-full border-b border-gray-300"></div>
        </div>
        <div className="relative flex justify-center">
          <span className="px-4 bg-secondary">Or</span>
        </div>
      </div>
      <a
        href={FACEBOOK_URL}
        className="flex justify-center py-3 rounded-md border mt-4 text-primary
bg-color2"
      >
        <img
          src={facebookLogo}
          alt="facebook"
          className="w-[24px] h-[24px] mr-1"
        />
        Login with Facebook
      </a>
      <p className="text-center mt-4">
        Don't have account?{" "}
        <Link to="/register" className="text-color1">
          register
        </Link>
      </p>
    </form>
  </div>
</MainLayout>
);

```

Figure 18: Screenshot of LoginPage.jsx

15) Construct the application which implements the selected API in Task 2.

Using Facebook Login API

Step 1: Go to <https://developers.facebook.com/> and login

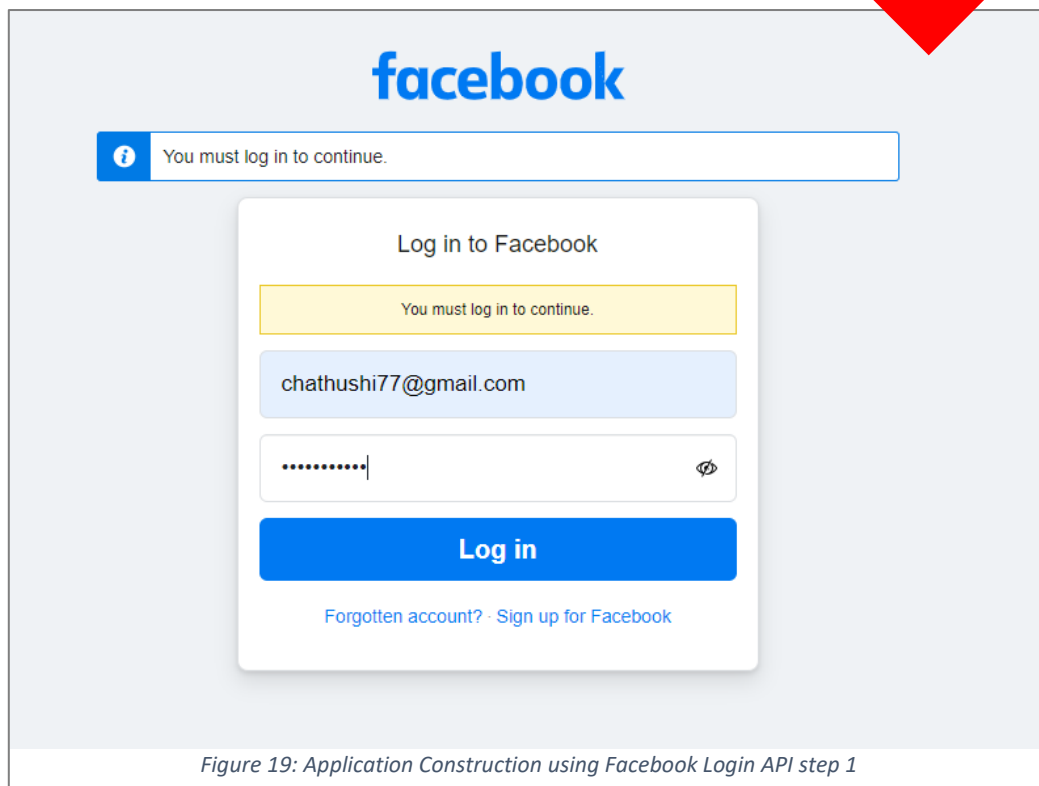
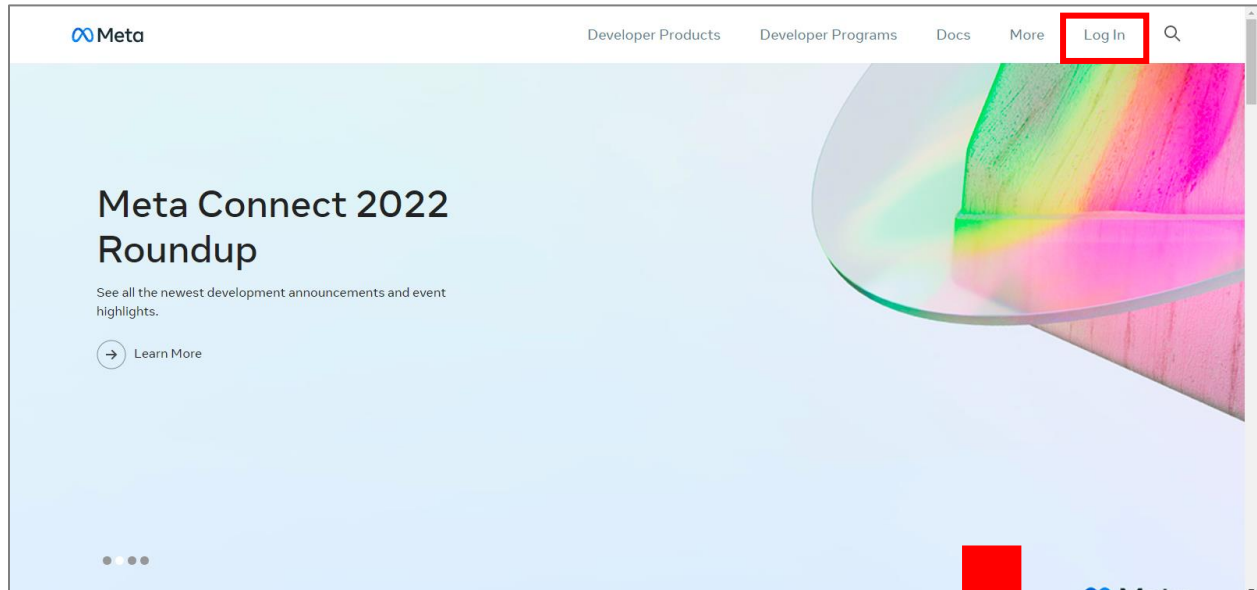


Figure 19: Application Construction using Facebook Login API step 1

Step 2: After login, you can see the following interface. You can click create app button and create app

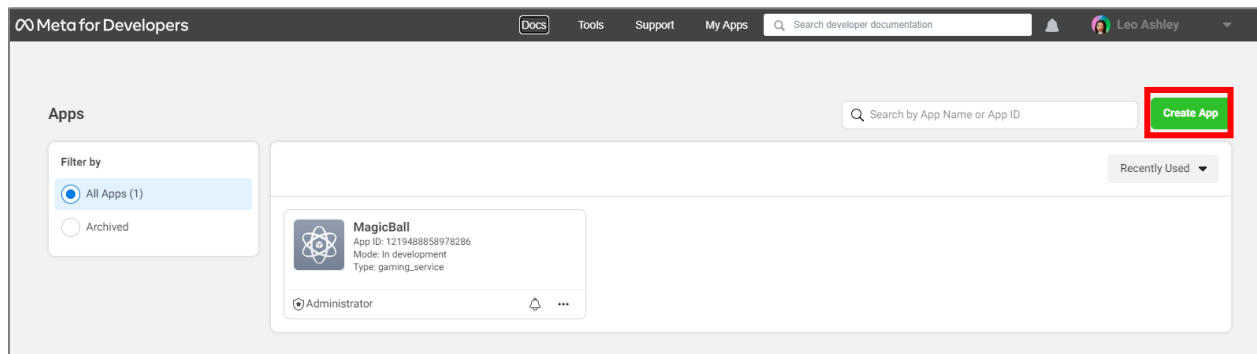


Figure 20: Application Construction using Facebook Login API step 2

Step 3: Select type as “Gaming” and click next

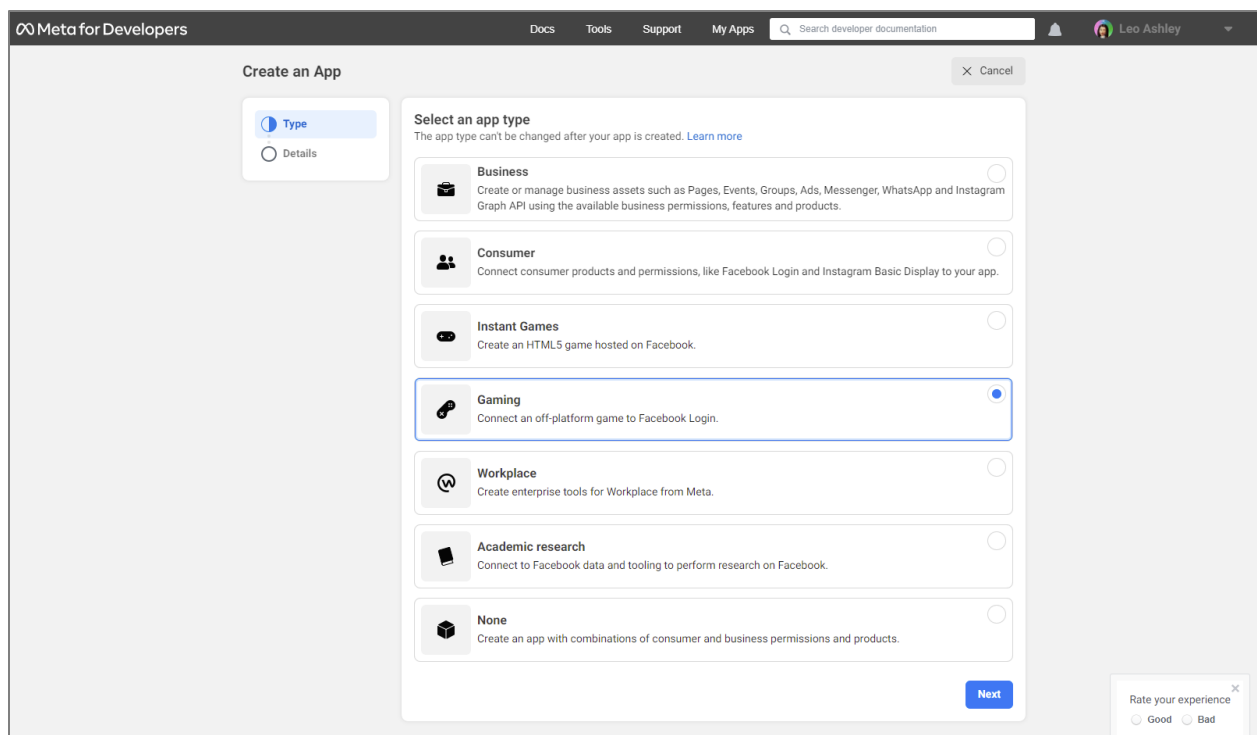
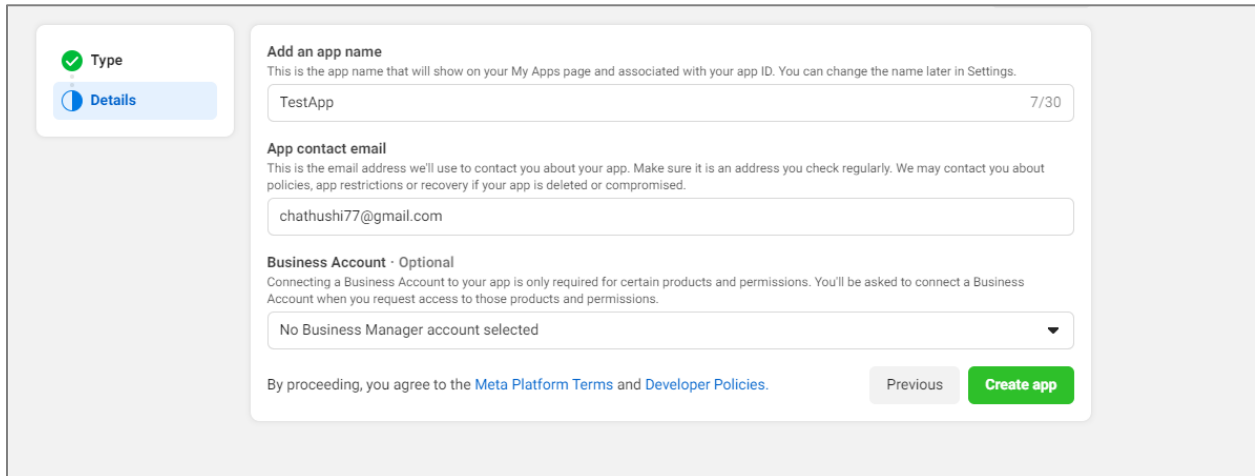


Figure 21: Application Construction using Facebook Login API step 3

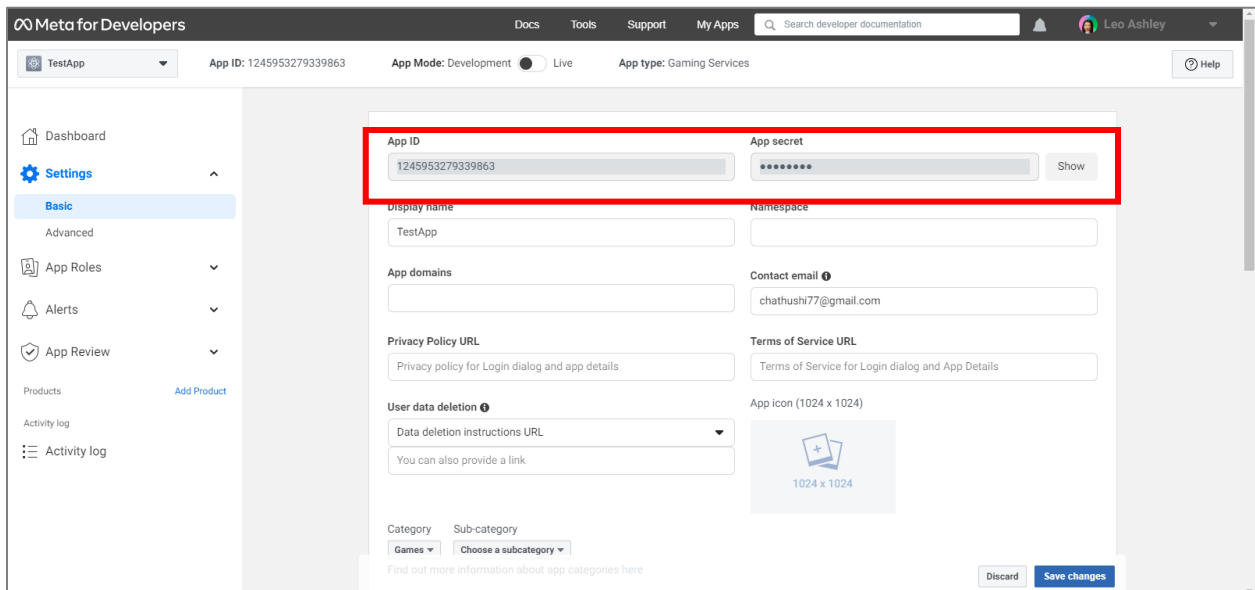
Step 4: Give a name for app and contact email and click create app



The screenshot shows the 'Add an app name' form in the Meta for Developers console. On the left, there are two tabs: 'Type' (selected) and 'Details'. The main form area has three sections: 'Add an app name' with a text input containing 'TestApp' and a character count '7/30'; 'App contact email' with a text input containing 'chathushi77@gmail.com'; and 'Business Account - Optional' with a dropdown menu showing 'No Business Manager account selected'. At the bottom, there is a 'Previous' button and a green 'Create app' button. A note at the bottom states: 'By proceeding, you agree to the [Meta Platform Terms](#) and [Developer Policies](#)'.

Figure 22: Application Construction using Facebook Login API step 4

Step 5: Copy the app ID and app secret for us and paste it on *application.yml*



The screenshot shows the 'TestApp' settings page in the Meta for Developers console. The top bar includes the 'Meta for Developers' logo, navigation links (Docs, Tools, Support, My Apps), a search bar, and a user profile 'Leo Ashley'. The main content area is divided into a left sidebar with navigation links (Dashboard, Settings, Basic, Advanced, App Roles, Alerts, App Review, Products, Activity log) and a main form area. The 'App ID' is '1245953279339863' and the 'App Mode' is 'Development'. The 'App type' is 'Gaming Services'. The 'App ID' and 'App secret' fields are highlighted with a red box. The 'App secret' is masked with asterisks and has a 'Show' button. Other fields include 'Display name' (TestApp), 'Namespace', 'App domains', 'Contact email' (chathushi77@gmail.com), 'Privacy Policy URL', 'Terms of Service URL', 'User data deletion', and 'App icon' (1024 x 1024). At the bottom, there are 'Discard' and 'Save changes' buttons.

Figure 23: Application Construction using Facebook Login API step 5

Application.yml

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/api_know
    username: root
    password: divergent77

  jpa:
    show-sql: true
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL8Dialect
  security:
    oauth2:
      client:
        registration:
          facebook:
            clientId: 695457215366736
            clientSecret: f316dc1ab0d7f50f6c323aac9ddb84eb
            redirectUri: http://localhost:8080/oauth2/callback/facebook
            scope:
              - email
              - public_profile
        provider:
          facebook:
            authorizationUri: https://www.facebook.com/v3.0/dialog/oauth
            tokenUri: https://graph.facebook.com/v3.0/oauth/access_token
            userInfoUri: https://graph.facebook.com/v3.0/me?
      fields=id,name,email,picture.width(250).height(250)

  app:
    auth:
      tokenSecret: 9A95527AD63560CDC6F7556D5411A
      tokenExpirationMsec: 86400000
    oauth2:
      authorizedRedirectUris:
        - http://localhost:3000/oauth2/redirect
```

Figure 24: Application Construction using Facebook Login API (Application.yml)

Task 4

16) Implement white Box testing for the developed API of your Application

16.1 What is white box testing?

White box testing is a testing approach wherein software program's inner shape, design, and coding are tested to confirm enter-output flow and improve design, usability, and safety. In white box testing, code is visible to testers, so it's also known as clean container checking out, Open field testing, transparent container testing, Code-primarily based testing, and Glass box testing.

it's miles one in all two parts of the box trying out technique to software testing. Its counterpart, Blackbox testing, includes testing from an outside or quit-consumer attitude. alternatively, White box testing in software program engineering is primarily based on the internal workings of an software and revolves round inner testing.

16.2 Why we use white box testing?

White box testing ensures,

- Business requirement are satisfied
- Each entity of tech is perfectly integrated together
- Enable to search invisible bugs
- Ensure greater user experience

Testing APIs with Postman

Register

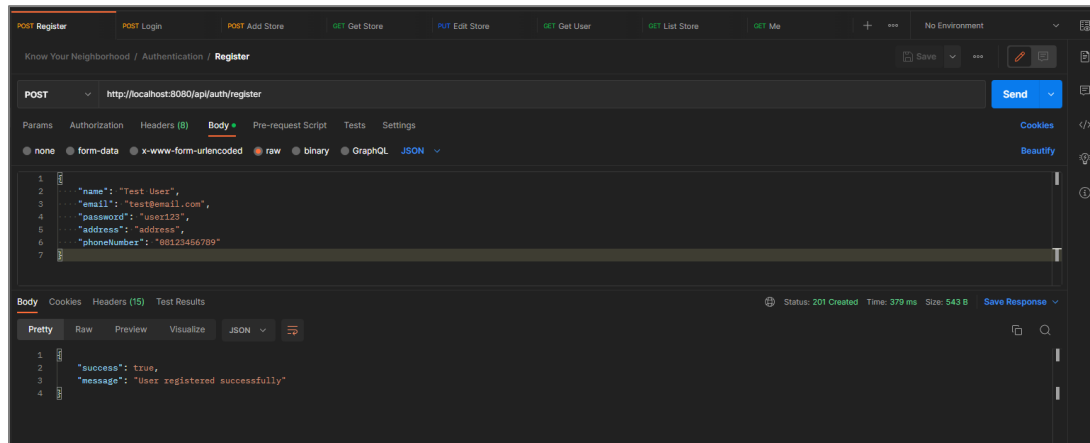


Figure 25: Postman Testing - Registration

Login

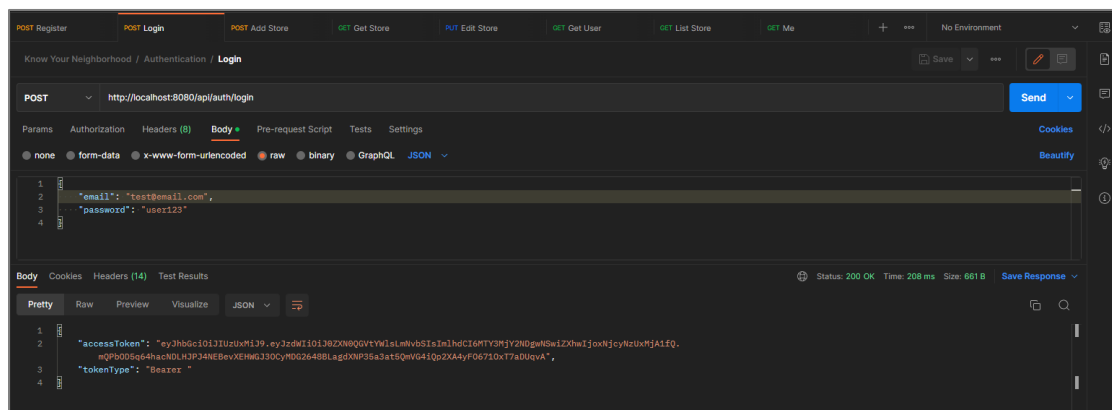


Figure 26:Postman Testing - Login

Get User Login

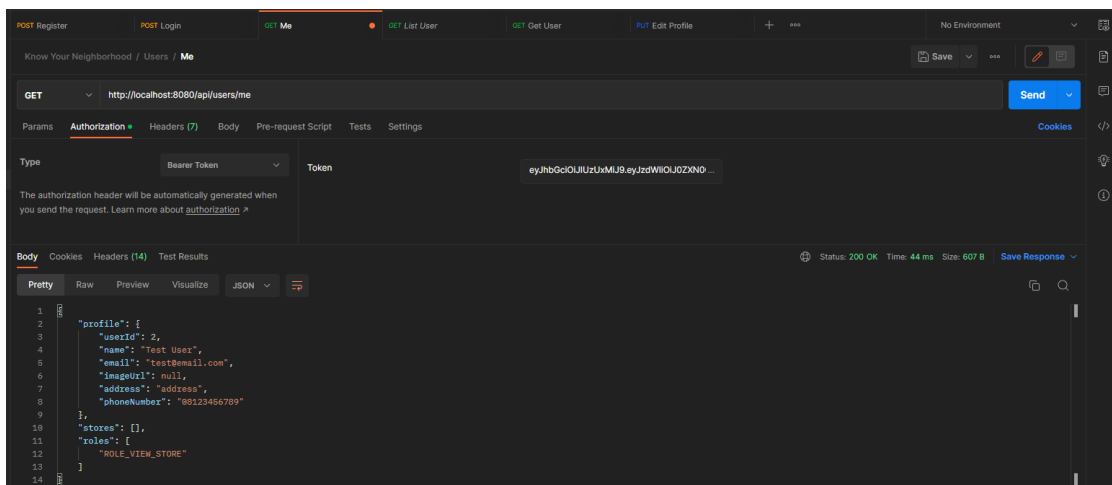


Figure 27:Postman Testing - Get user Login

- 17) Conduct Black Box testing (UAT testing) of your developed application and show the evidence for each test case.

17.1 What is black box testing?

Black box testing involves testing a machine with no prior knowledge of its internal workings. A tester gives input, and observes the output generated through the gadget underneath take a look at. This makes it possible to identify how the system responds to predicted and unexpected person movements, its response time, usability troubles and reliability issues.

Black container testing is an effective testing method because it physical activities a gadget give up-to-stop. much like cease-users “don’t care” how a device is coded or architected, and expect to acquire the perfect response to their requests, a tester can simulate consumer activity and notice if the system gives you on its promises. alongside the manner, a black field take a look at evaluates all relevant subsystems, which includes UI/UX, net server or application server, database, dependencies, and incorporated systems.

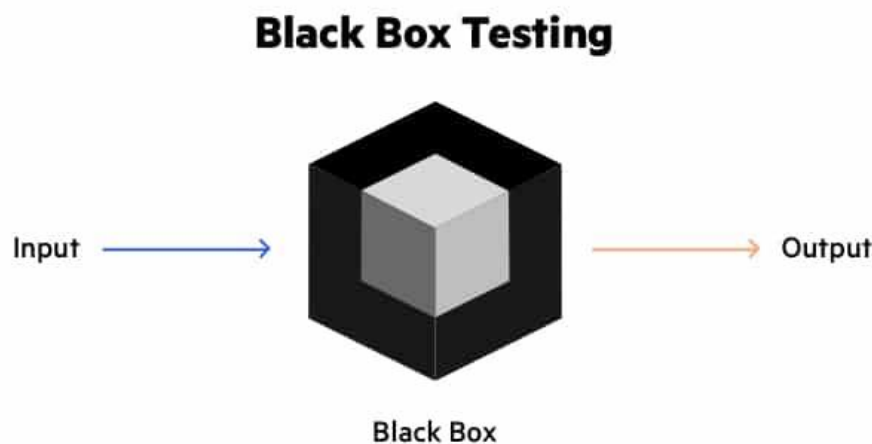


Figure 28:Black box testing

17.2 Benefits of black box testing

- The tester doesn't need any technical expertise to check the machine. It is essential to recognize the user's perspective.
- Testing is achieved after development, and both the sports are independent of every other.
- It works for a greater giant insurance which is typically neglected out by way of testers as they fail to see the larger photograph of the software.
- Test cases can be generated before improvement and right after specification.
- Black box test method is near agile.

Registration

Test Scenario	Registration
TS001	
Test Cases	Registration in the Know Your Neighborhood Website
TC001	Validating data that is inpputed into registration form
TC002	After user submitting correct data it should show a success message

Test case	Scenario	Preconditions	Execution steps	Test Data	Expected Result	Actual Result	Pass/Fail	Reference Evidence
TC01	Ensure that users can register on the KYN website	the website has been designed and developed	<ul style="list-style-type: none"> Go to KYN Website Go to login page Click on the register button Fill in all input data Click on Register button 	email = "_____" Password = 123456 Name = Chathu Address = Kandy Phone number = 0123456789	If user enter blank data It should throw an error .	As expected,	PASS	TS001 – TC01
TC02	Ensure that users can register on the KYN website	the website has been designed and developed	<ul style="list-style-type: none"> Go to KYN Website Go to login page Click on the register button Fill in all input data Click on Register button 	email = chathushi77@gmail.com Password = 1234567 Name = Chathu Address = Kandy Phone number = 0123456789	Show success message after submitting data .	As expected,	PASS	TS001 – TC02

Fig: TS001 – TC01

The screenshot shows the KYN website's Register form. The form is centered on a dark background. The top navigation bar includes the KYN logo, links for Home, Stores, About, and Contact, and buttons for Login and Register. The Register form contains the following fields and values:

- Email***: Enter your email (Error: Email is required)
- Password***: (Error: Password is required)
- Name***: Chathushi
- Address**: Kandy
- Phone Number**: 0123456789

A Register button is located at the bottom of the form. The footer text reads: "Created by Chathushi Jayarathna © 2023 Copyright | All Rights Reserved".

Fig: TS001 – TC02

The screenshot shows the KYN website's Register form with the following data entered:

- Email***: chathushi77@gmail.com
- Password***: (Password is masked with asterisks)
- Name***: Chathushi
- Address**: Colombo
- Phone Number**: 0123456789

The Register button is visible at the bottom of the form. The footer text reads: "Created by Chathushi Jayarathna © 2023 Copyright | All Rights Reserved".

Register

User registered successfully



Email*

Password*

Name*

Address

Phone Number

Login

Test Scenario	Login
TS002	
Test Cases	Login in Know Your Neighborhood Website
TC001	Login with local account
TC002	Login with facebook API

Test case	Scenario	Preconditions	Execution steps	Test Data	Expected Result	Actual Result	Pass/Fail	Reference Evidence
TC01	Ensure that users can register on the KYN website	the website has been designed and developed	<ul style="list-style-type: none"> Go to KYN Website Go to login oage Click on the register button Fill in all input data Click on login button 	Email = chathushi77@email.com Password = 123456	After user entering valid data, user should be redirected to their profile page .	As expected,	PASS	TS002 – TC01
TC0	Ensure that users can register on the KYN website	the website has been designed and developed	<ul style="list-style-type: none"> Go to KYN Website Go to login oage Click on the register button Fill in all input data Click on login button 	“Valid facebook account”0123456789	After user entering valid data, user should be redirected to their profile page	As expected,	PASS	TS002-TC01

Fig: TS002 – TC01

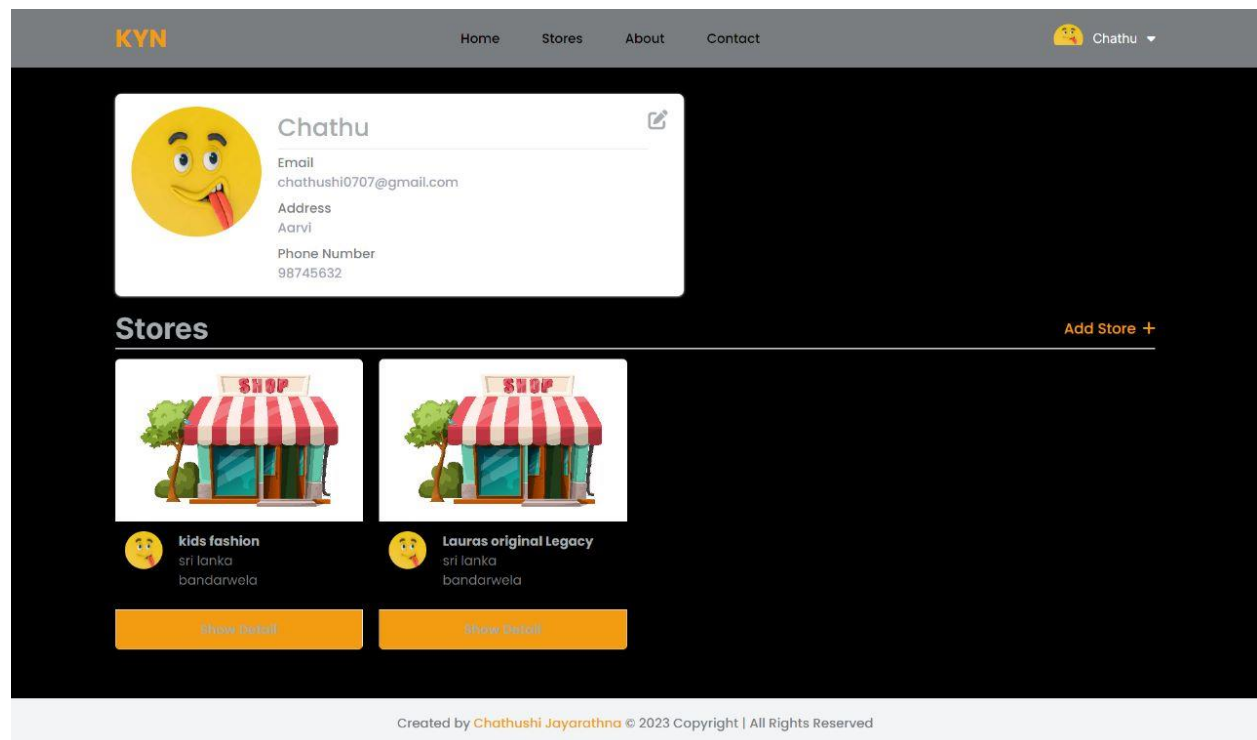
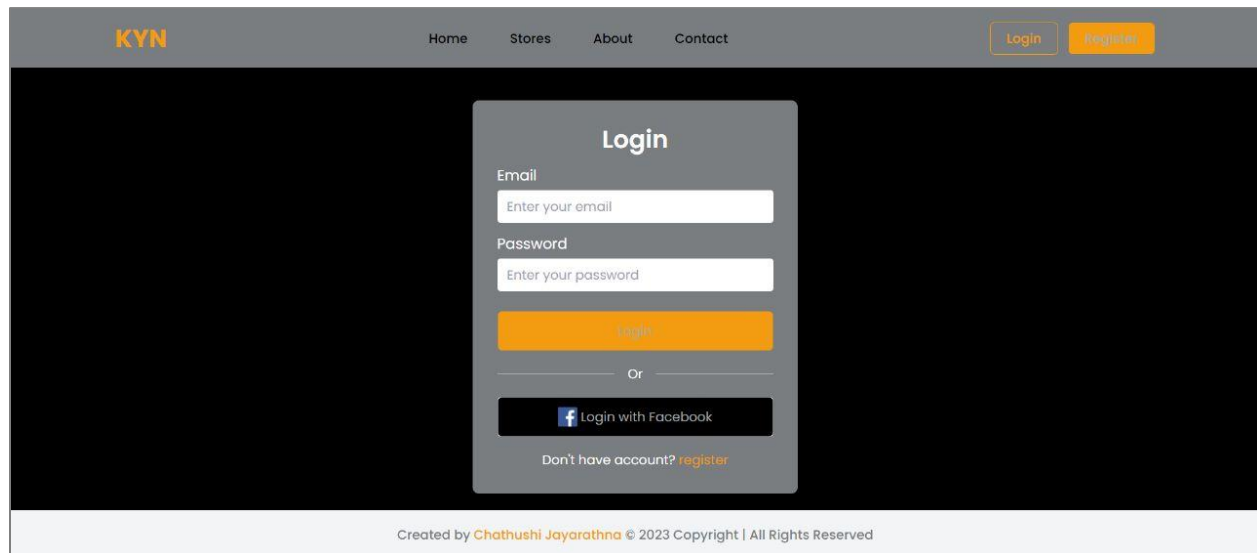


Fig:TS002- TC02

facebook

Create New Account

Log in to Facebook

Email address or phone number

Password

Log in

Forgotten account?

or

Create New Account

Not now

Task 5

- 18) Review your developed API, and identify the strength and weaknesses of API.

REST API

Strengths of REST API

- **Flexible across languages and frameworks**

One obvious reason why the rest API has been so successful is that it really works across languages and frameworks. No matter how developers build their programs, a rest API will work for them.

- **High interpretability**

The fact that rest APIs are URL-primarily based, in which every URL gives get right of entry to a certain amount of data, renders rest APIs extraordinarily easy to interpret. All a third-birthday party developer calls for is a URL, knowledge of what statistics they will discover there, and knowledge of the way they are able to engage with that information.

- **Server-side logic**

For an agency constructing a rest API, its URL-pushed nature is extraordinarily simple to work with. The server-aspect builders actually define a URL in which they want positive information to be seen, write common sense for the way to cope with every form of request, and the endpoint is created. However, as an employer's API scales in utilization, identifying the ideal manner to structure endpoints and the statistics they offer increases in complexity, a relative weakness of this architecture, transitioning to our next phase.

Weakness of REST API

- They may be less at ease than different types of APIs, as they rely upon the underlying delivery protocol (HTTP) to offer protection.
- They may be vulnerable to mistakes if no longer implemented well, as they rely on clients to offer accurate entries.
- They can be less efficient than different kinds of APIs, as they require extra overhead (e.g. parsing JSON or XML) to manner facts.
- Few use instances may additionally require extra complex features than the rest can provide.

19) Provide data security report of your developed application.

Spring Security

Spring security is a powerful and incredibly customizable authentication and access-manage framework. it is the de-facto standard for securing Spring-primarily based applications.

Spring security is a framework that specializes in offering each authentication and authorization to Java packages. like several Spring tasks, the real power of Spring security is discovered in how effortlessly it is able to be prolonged to fulfill custom requirements

Features

- Comprehensive and extensible support for both Authentication and Authorization
- Protection against attacks like session fixation, clickjacking, cross site request forgery, etc.
- Servlet API integration

Authentication with JSON Web Token (JWT)

Authentication with JSON net Token (JWT) is a famous technique for securing web packages. There are several motives why this project chooses to apply JWT for authentication:

- JWTs are self-contained: A JWT contains all the records necessary to authenticate a single user, compact token. This eliminates the want to shop authentication statistics in a consultation or at the server, which can simplify the implementation of the application.
- JWTs are at ease: JWTs are signed with a secret key, which makes it difficult for attackers to forge them or tamper with their contents.
- JWTs are broadly supported: JWT is a trend that is supported with the aid of a huge variety of platforms and libraries, making it clean to combine into your application.

Security with OAuth2

OAuth2 allows users to grant third-party applications access to their resources without sharing their login credentials. This can help to prevent unauthorized access to user accounts and data.

Task 6

20) Developed Pages

Home Page

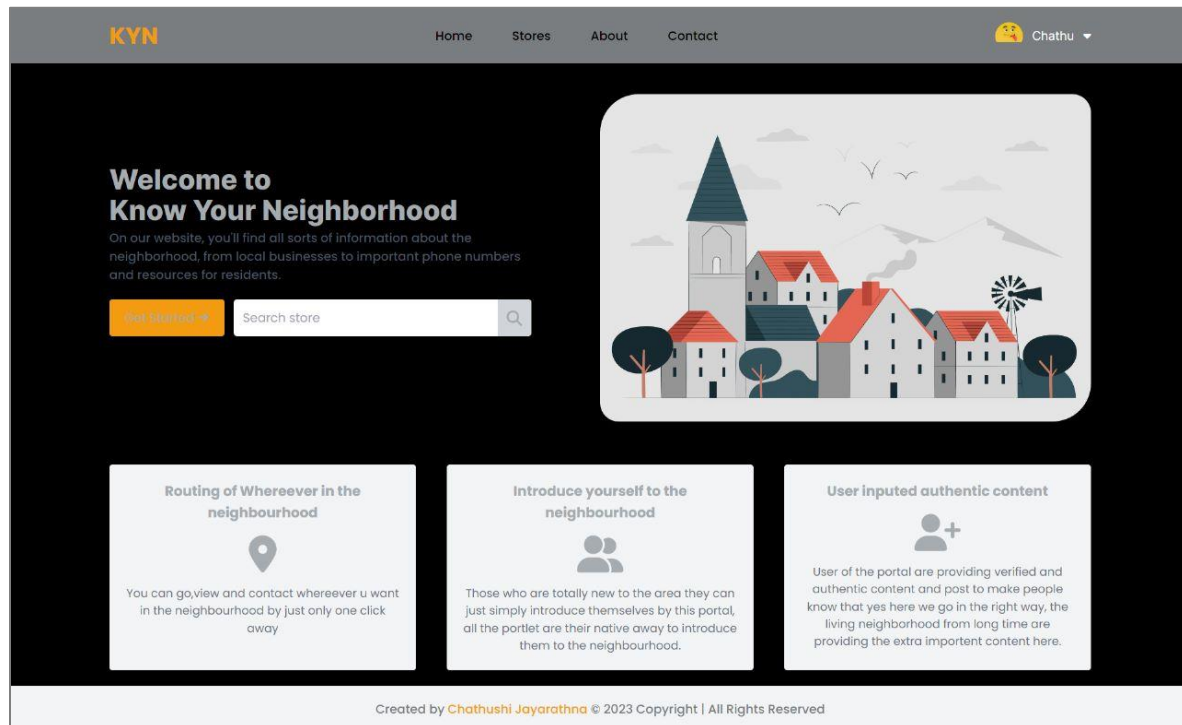


Figure 29:Screenshot of Home Page

Login Page

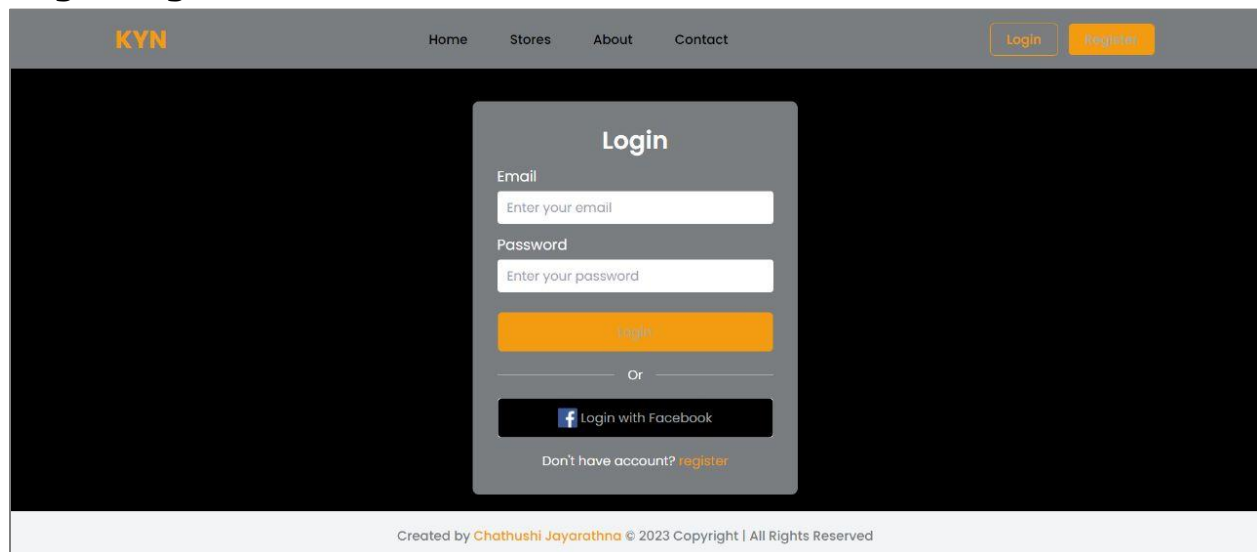
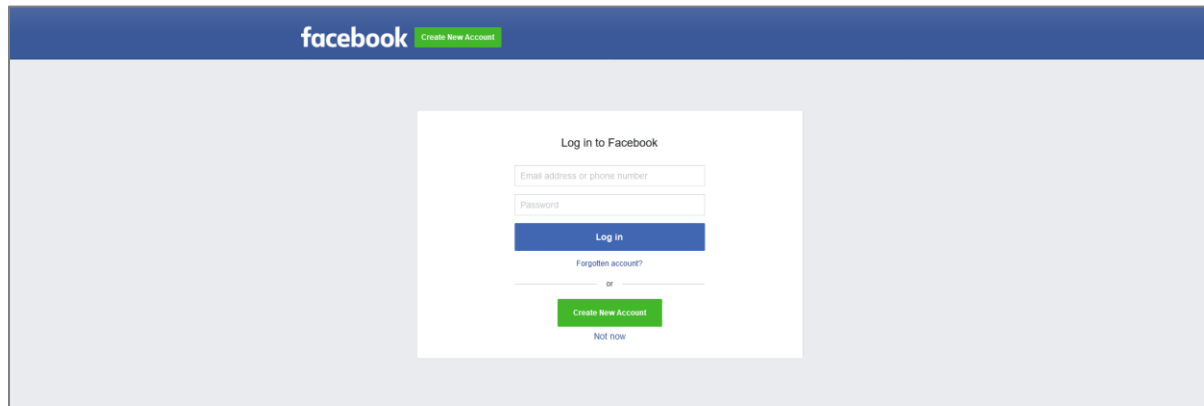


Figure 30:Screenshot of Login Page

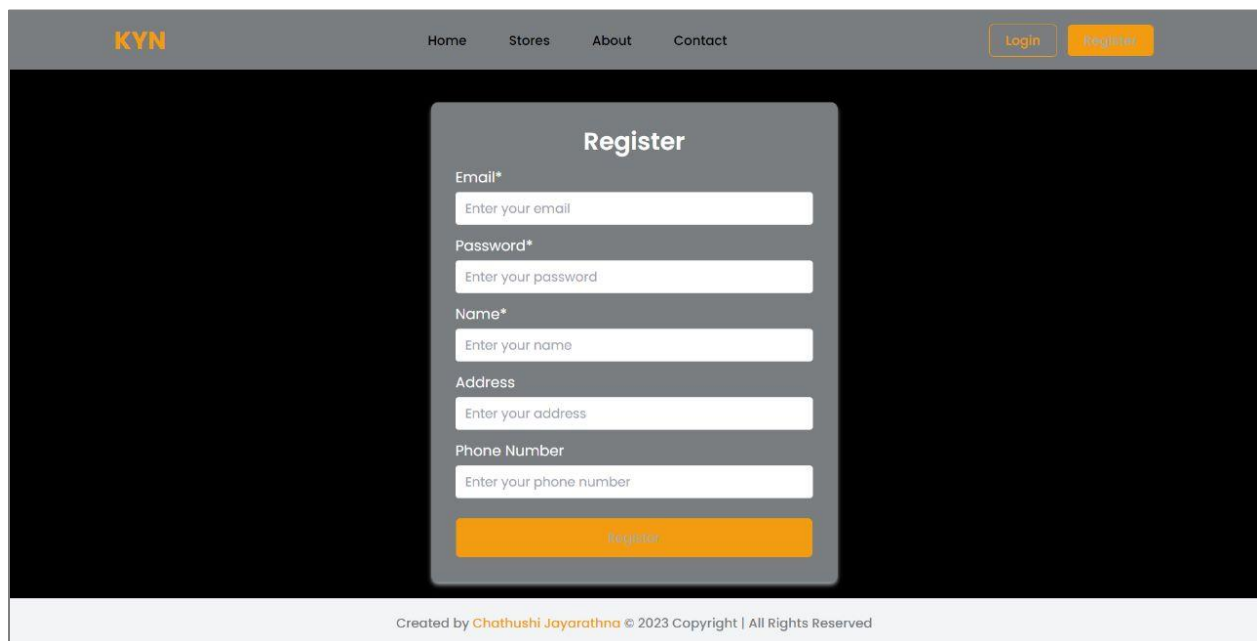
Facebook Login



The screenshot shows the Facebook login interface. At the top, there is a blue header with the Facebook logo and a green button labeled "Create New Account". Below the header, the main content area is light gray. In the center, there is a white box titled "Log in to Facebook". Inside this box, there are two input fields: "Email address or phone number" and "Password". Below these fields is a blue "Log in" button. Underneath the button, there is a link for "Forgotten account?". Below this, there is a horizontal line with "or" in the center. At the bottom of the white box, there is a green "Create New Account" button and a link for "Not now".

Figure 31: Screenshot of Facebook login

Register



The screenshot shows the registration page for KYN. The header is dark gray with the KYN logo on the left and navigation links "Home", "Stores", "About", and "Contact" in the center. On the right side of the header, there are two orange buttons: "Login" and "Register". The main content area has a black background. In the center, there is a gray box titled "Register". Inside this box, there are five input fields: "Email*", "Password*", "Name*", "Address", and "Phone Number". Each field has a placeholder text: "Enter your email", "Enter your password", "Enter your name", "Enter your address", and "Enter your phone number". Below these fields is an orange "Register" button. At the bottom of the page, there is a light gray footer with the text "Created by Chathushi Jayarathna © 2023 Copyright | All Rights Reserved".

Figure 32: Screenshot of Registration

Stores

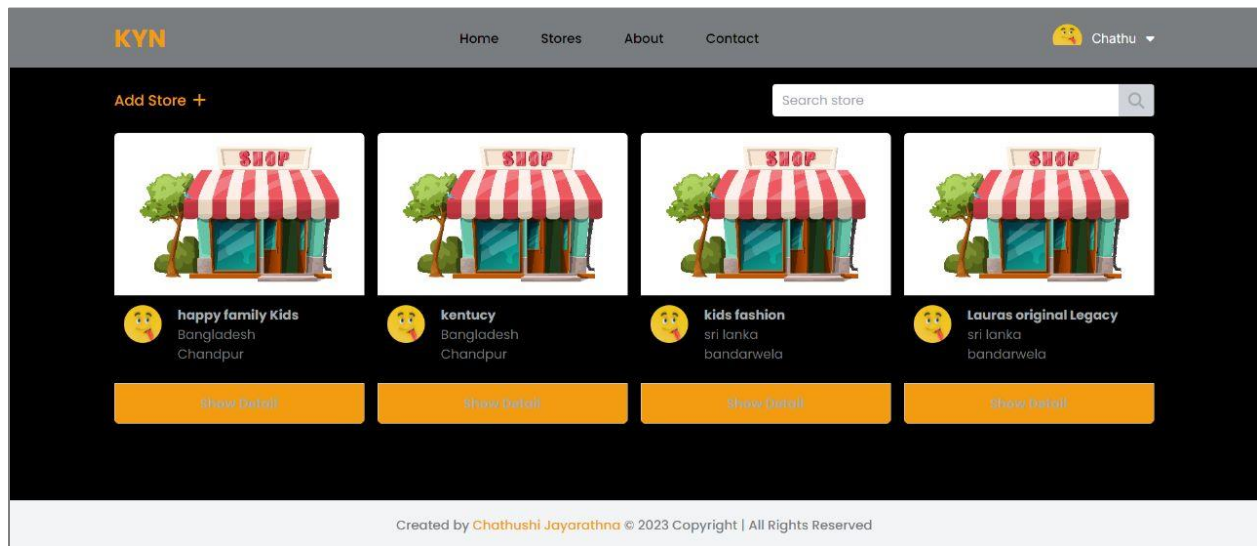


Figure 33: Screenshot of Stores

Store Detail

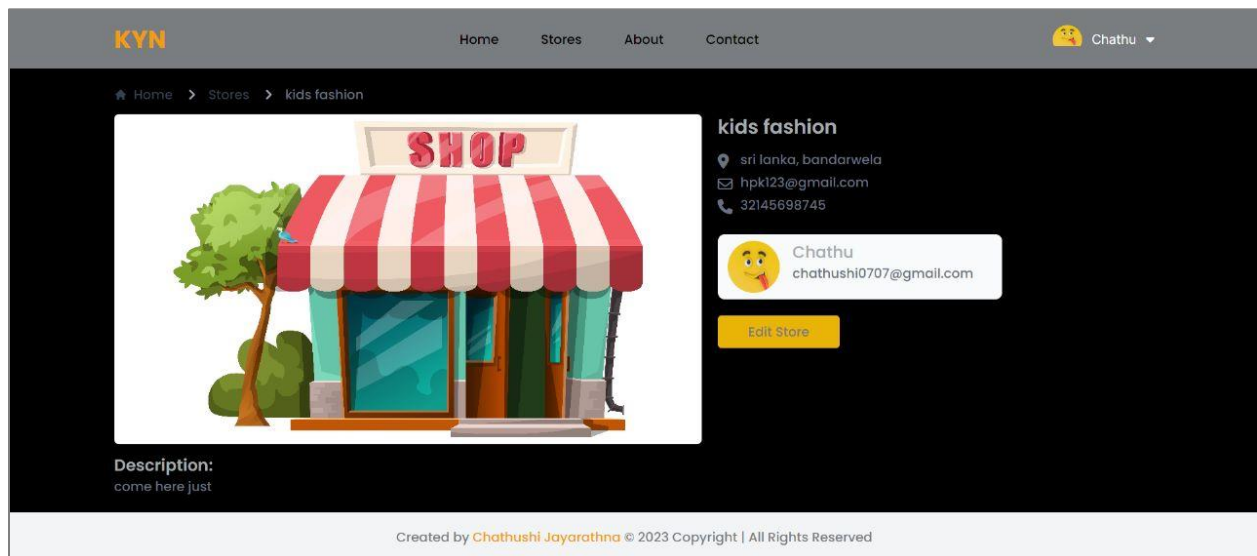


Figure 34: Screenshot of Screenshot Store Details

Add Store

The screenshot shows the 'Add Store' form within the KYN application. The header bar is dark gray with the 'KYN' logo in orange on the left and navigation links 'Home', 'Stores', 'About', and 'Contact' in the center. On the right, there is a user profile icon and the name 'Chathu' with a dropdown arrow. The main content area has a dark background. A light gray modal box is centered, titled 'Add Store'. It contains several input fields: 'Store Name*' (placeholder: 'Enter store name'), 'Country*' (placeholder: 'Enter country'), 'City*' (placeholder: 'Enter city'), 'Email*' (placeholder: 'Enter store email'), 'Phone Number*' (placeholder: 'Enter phone number'), and 'Description' (placeholder: 'Enter store description'). At the bottom of the modal is an orange button labeled 'Add Store'. The footer of the application is a light gray bar with the text 'Created by Chathushi Jayarathna © 2023 Copyright | All Rights Reserved'.

KYN Home Stores About Contact Chathu

Add Store

Store Name*
Enter store name

Country*
Enter country

City*
Enter city

Email*
Enter store email

Phone Number*
Enter phone number

Description
Enter store description

Add Store

Created by Chathushi Jayarathna © 2023 Copyright | All Rights Reserved

Figure 35: Screenshot of Add Stores

Edit Store

The screenshot shows the 'Edit Store Detail' form within the KYN application. The header bar is identical to the previous screenshot. The main content area has a dark background. A light gray modal box is centered, titled 'Edit Store Detail'. It contains the same input fields as the 'Add Store' form, but with pre-filled data: 'Store Name*' (Lauras original legacy), 'Country*' (sri lanka), 'City*' (bandarweila), 'Email*' (Lol@l23business.com), 'Phone Number*' (587964123), and 'Description' (join with us). At the bottom of the modal is an orange button labeled 'Save Edit'. The footer of the application is a light gray bar with the text 'Created by Chathushi Jayarathna © 2023 Copyright | All Rights Reserved'.

KYN Home Stores About Contact Chathu

Edit Store Detail

Store Name*
Lauras original legacy

Country*
sri lanka

City*
bandarweila

Email*
Lol@l23business.com

Phone Number*
587964123

Description
join with us

Save Edit

Created by Chathushi Jayarathna © 2023 Copyright | All Rights Reserved

Figure 36: Screenshot of Edit Store

Profile Page

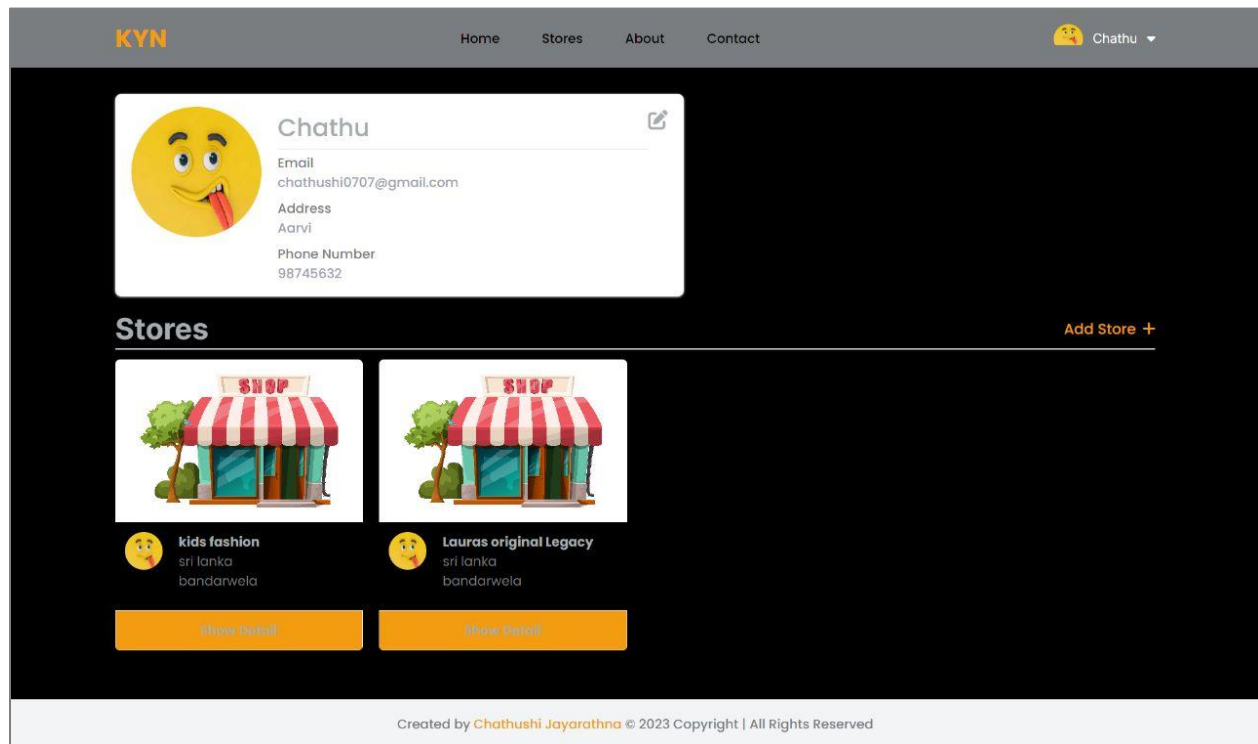


Figure 37: Screenshot of Profile Page

Edit Profile

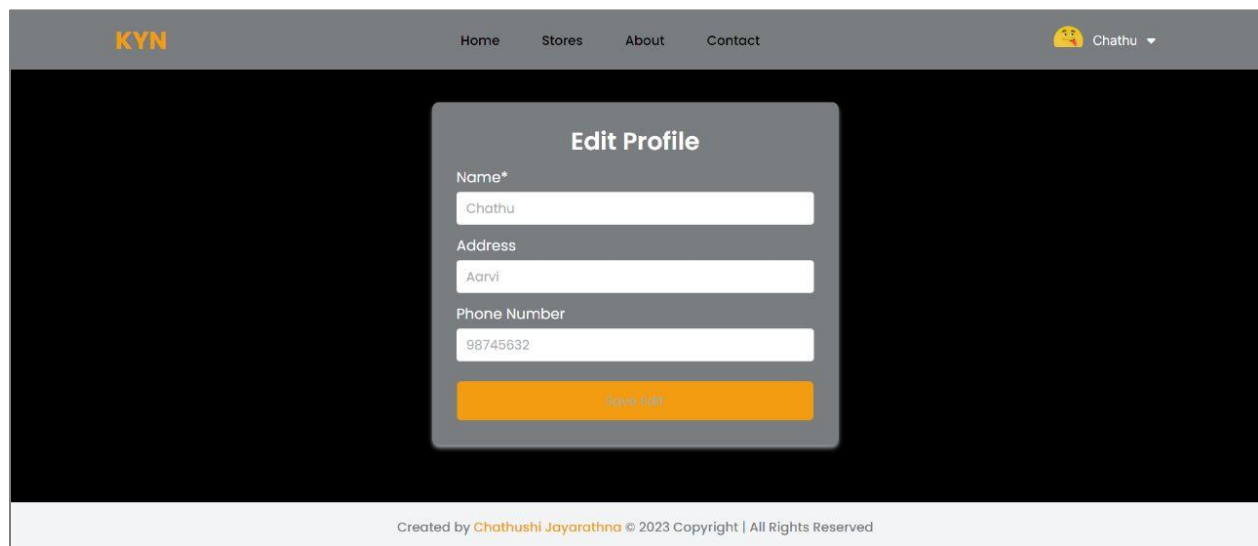


Figure 38: Screenshot of Edit Profile Page

About Us

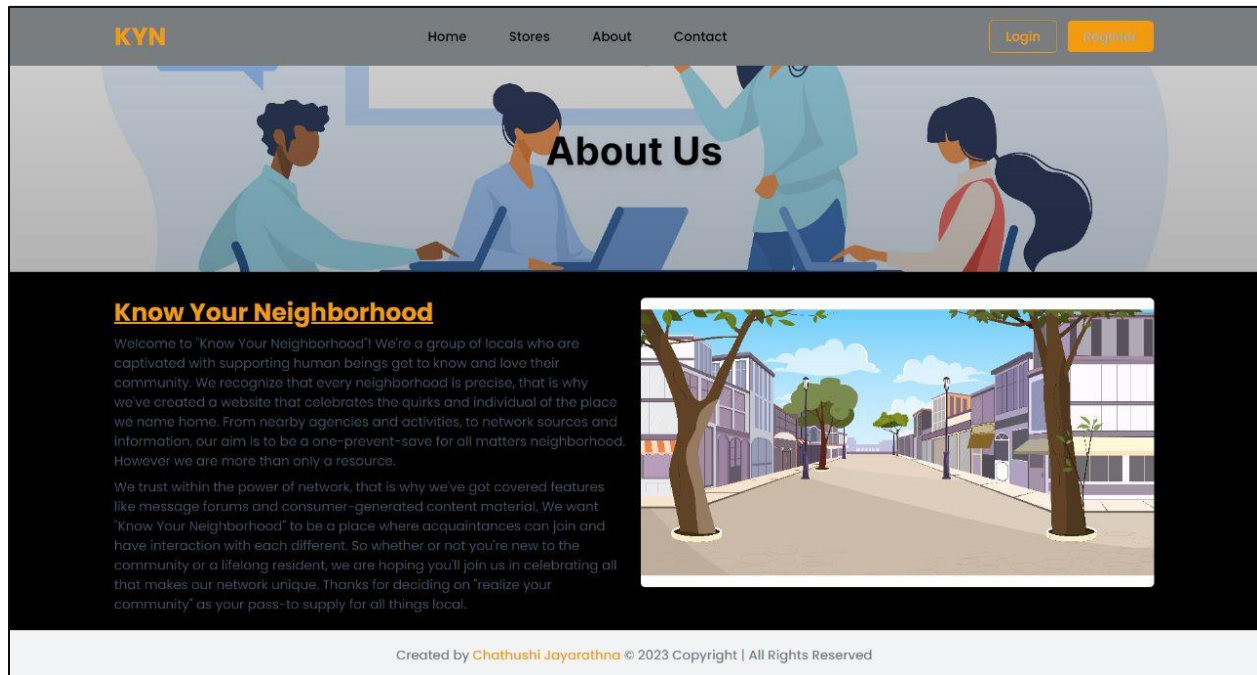


Figure 39: Screenshot of About Us Page

Contact Us

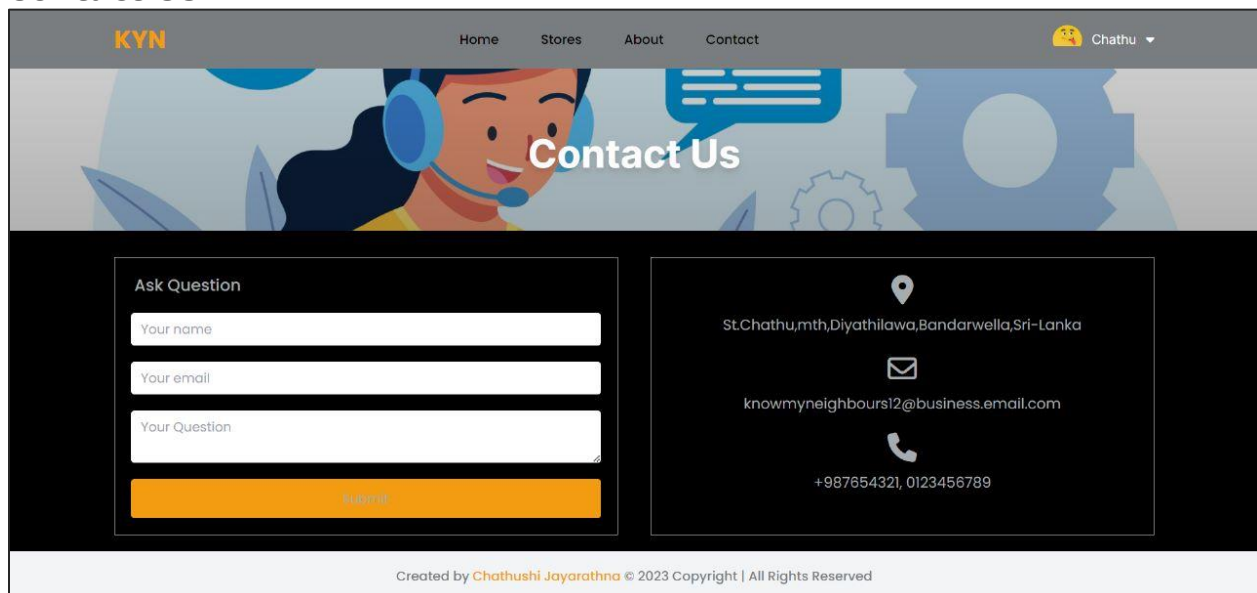


Figure 40: Screenshot of Contact Us Page