SCRATCH CARD EXCHANGE PORTAL

A MINI PROJECT REPORT

18CSC207J - ADVANCED PROGRAMMING PRACTICE

Submitted by

ANJALI JHA [RA2111003011873]
ROSELEEN GEORGE [RA2111003011877]
BHUVAN BEERA [RA2111003011907]

Under the guidance of

Dr. K .M. UMAHESHWARI

Assistant Professor, Department of Computer Science and Engineering

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"SCRATCH CARD EXCHANGE PORTAL"** is the bonafide work of **ANJALI JHA [RA2111003011873] ROSELEEN GEORGE [RA2111003011877] BHUVAN BEERA [RA2111003011907]**who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported here in does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

DATE: -

SIGNATURE

LAB INCHARGE

Dr. K .M. UMAHESHWARI

Assistant Professor Department of Computing Technologies

SRM Institute of Science and Technology

SIGNATURE

HEAD OF THE DEPARTMENT

Dr. M. PUSHPALATHA

Professor & Head

Department of Computing Technologies
SRM Institute of Science and Technology

TABLE OF CONTENTS

ABSTRACT

This project aims to develop an online portal where friends and students can buy, sell, or exchange scratch cards of various types using Python and Visual Studio Code. The portal will be built using the Django web framework, a popular and powerful tool for building web applications, which provides a range of features such as an ORM, routing, templating, and more.

The portal will be designed to be user-friendly and easy to navigate, with a responsive and visually appealing theme. The portal will also feature a payment system that will allow users to securely purchase scratch cards, as well as a user account management system that will enable users to create accounts, view their transactions history, and manage their profile information.

To build the portal, we will use the Visual Studio Code integrated development environment (IDE), which provides an intuitive and customisable interface for writing, debugging, and testing Python code. We will also use a range of Python libraries and tools, such as Django Rest Framework, which will allow us to create RESTful APIs for the portal, and Stripe, which will provide a secure payment processing system.

Once the portal is developed, we will thoroughly test it to ensure it functions as intended, is secure, and provides a smooth user experience. We will also gather user feedback to identify any issues or areas for improvement.

The final report for this project will detail the project goals, requirements, platform, tools, challenges faced, and user feedback. The report will also include screenshots and videos to illustrate the site's functionality, as well as code snippets to showcase the Python and Django development process. This project aims to demonstrate the power and versatility of Python and Django for building complex and dynamic web applications, as well as the benefits of using Visual Studio Code as an IDE for Python development.

INTRODUCTION

Scratch cards are a popular and convenient way to access a variety of products and services, from mobile phone credit to online gaming. However, many people find themselves with unused or unwanted scratch cards, while others struggle to find specific types of cards they need. To address this issue, we propose to develop an online portal where friends and students can buy, sell, or exchange scratch cards of various types.

Our project will be built using Python and the Django web framework, a powerful and versatile tool for building web applications. We will also use the Visual Studio Code IDE, which provides an intuitive and customizable interface for developing and testing Python code. With these tools, we aim to create a user-friendly and secure online portal that meets the needs of students and friends who want to trade their scratch cards.

The portal will feature a range of functionalities, including a payment processing system that will allow users to securely purchase scratch cards, as well as a user account management system that will enable users to create accounts, view their transactions history, and manage their profile information. We will also aim to make the portal mobile-friendly and responsive, so that users can easily access it from their smartphones and tablets.

Overall, our project aims to demonstrate the power and versatility of Python and Django for building complex and dynamic web applications, as well as the benefits of using Visual Studio Code as an IDE for Python development. By creating a user-friendly and secure platform for buying, selling, and exchanging scratch cards, we hope to meet the needs of students and friends, while showcasing the potential of modern web development tools and frameworks.

LITERATURE SURVEY

The use of scratch cards for accessing various services and products has become increasingly popular in recent years. As a result, several online platforms have emerged to cater to this market, offering services such as scratch card trading and exchange. One such platform is Card tonic, a Nigerian-based online platform that allows users to buy, sell, and exchange various types of gift cards, including mobile recharge cards and gaming cards. Card tonic has become popular in Nigeria due to its user-friendly interface and secure payment system, as well as its wide range of supported gift cards.

Another platform is Raise, an online marketplace that allows users to buy and sell gift cards from various retailers, such as Amazon and Walmart. Raise provides a secure payment system and offers a guarantee on all transactions, ensuring that users are protected from fraud and scams.
In the context of Python and Django development, several resources are available to help developers build web applications. The official Django documentation provides a comprehensive guide to using the framework, including tutorials, API references, and best practices. The Django Girls tutorial is also a popular resource for beginners, providing a step-by-step guide to building a simple web application using Django.

In addition, several libraries and tools are available to extend the functionality of Python and Django, such as Django Rest Framework, which allows developers to create RESTful APIs for their applications, and Stripe, which provides a secure payment processing system.

Overall, the literature survey highlights the growing popularity of online platforms for scratch card trading and exchange, as well as the availability of resources and tools for Python and Django development. By leveraging these resources and building a userfriendly and secure platform, we aim to contribute to this growing market while showcasing the potential of modern web development tools and frameworks.

SYSTEM ARCHITECTURE AND DESIGN

The online scratch card portal will be designed as a web application built using the Python programming language and the Django web framework. The application will be hosted on a web server and accessed by users through their web browser. The system architecture will consist of three main components: the front-end, the backend, and the database.

The front-end component will be responsible for presenting the user interface to the user and handling user input. It will be built using HTML, CSS, and JavaScript, and will be designed to be mobile-friendly and responsive. We will use a modern front-end framework such as React or Vue.js to create a smooth and interactive user experience.

The back-end component will be responsible for handling the business logic of the application, such as processing payments, managing user accounts, and handling scratch card transactions. It will be built using Python and the Django web framework, which provides a robust and scalable platform for building web applications. We will use the Django Rest Framework to create a RESTful API for communicating between the front-end and back-end components.

The database component will be responsible for storing and retrieving data related to scratch cards, users, and transactions. We will use a relational database such as MySQL or PostgreSQL, which integrates seamlessly with Django's ORM (Object-Relational Mapping) system.

To ensure the security of the application, we will implement various security measures such as using HTTPS encryption for all communication, hashing user passwords before storing them in the database, and implementing a secure payment processing system such as Stripe.
Finally, we will deploy the application to a cloud-based web server such as Amazon Web Services (AWS) or Google Cloud Platform (GCP), which provides scalable and reliable hosting solutions for web applications.

Overall, the system architecture and design will be optimized for scalability, security, and user experience, leveraging the power and versatility of Python and Django to build a modern and user-friendly scratch card trading and exchange platform.

METHODOLOGY

Requirements gathering: We will gather requirements from stakeholders and users to determine the features and functionalities required for the scratch card portal. This will include features such as user account management, scratch card listing and search, and secure payment processing.

System design: We will design the system architecture and database schema based on the requirements gathered in step 1. This will involve selecting appropriate technologies and frameworks for each component of the system.
Front-end development: We will develop the front-end component of the application using a modern front-end framework such as React or Vue.js. This will involve creating user interfaces for scratch card listing and search, user account management, and payment processing.

Back-end development: We will develop the back-end component of the application using Python and the Django web framework. This will involve implementing business logic for scratch card transactions, user authentication and authorisation, and payment processing.

Database development: We will design and develop the database component of the application using a relational database such as MySQL or PostgreSQL. This will involve creating database schema, optimising query performance, and implementing data access logic using Django's ORM system.
Integration and testing: We will integrate the front-end, back-end, and database components of the system and perform rigorous testing to ensure that the application functions as expected. This will include testing for functionality, performance, and security.

Deployment: We will deploy the application to a cloud-based web server such as AWS or GCP, configure the server for scalability and reliability, and ensure that the application is accessible to users.
Maintenance and support: We will provide ongoing maintenance and support for the application, including bug fixes, security updates, and feature enhancements based on user feedback.

Overall, this methodology will allow us to build a robust, secure, and user-friendly online scratch card portal using Python and Django, leveraging modern web development frameworks and technologies

CODING AND TESTING

Front-end coding: The front-end component of the application will be developed using a modern front-end framework such as React or Vue.js. The front-end code will be written in JavaScript, HTML, and CSS, and will be responsible for creating the user interface and handling user interactions. The code will be tested using automated testing frameworks such as Jest or Enzyme.
Back-end coding: The back-end component of the application will be developed using Python and the Django web framework. The back-end code will be responsible for handling business logic such as processing payments, managing user accounts, and handling scratch card transactions. The code will be tested using automated testing frameworks such as Django's built-in testing framework or pytest.

Database coding: The database component of the application will be developed using a relational database such as MySQL or PostgreSQL. The database schema will be designed using Django's ORM system, and data access logic will be implemented in the back-end code. The database code will be tested using automated testing frameworks such as pytest.

Integration testing: Once the front-end, back-end, and database components have been developed, they will be integrated and tested as a complete system. This will involve testing the end-to-end functionality of the application, including user authentication, scratch card listing and search, payment processing, and transaction management.

User acceptance testing: Once the application has been developed and integrated, it will be subjected to user acceptance testing to ensure that it meets the requirements and expectations of users. This will involve soliciting feedback from a sample of users and making any necessary changes based on their feedback.

Performance testing: The application will be subjected to performance testing to ensure that it can handle a high volume of users and transactions. This will involve simulating a high volume of user traffic and monitoring the application's response time and resource utilisation.

Security testing: The application will be subjected to security testing to identify and address potential vulnerabilities such as SQL injection, cross-site scripting, and session hijacking. This will involve using automated testing tools such as OWASP ZAP and manual testing by security experts.

Overall, the coding and testing process will involve a combination of automated and manual testing to ensure that the application is robust, secure, and user-friendly, and meets the requirements and expectations of users.

Python code for creating a user model in Django:

```
from django.db import models from django.contrib.auth.models
import AbstractUser class
CustomUser(AbstractUser):       phone_number = models.CharField(max_length=10,
blank=True, null=True)     address = models.CharField(max_length=100, blank=True,
null=True)     profile_image = models.ImageField(upload_to='profile_images/', blank=True,
null=True)
   def __str__(self):         return
self.username
```

Python code for creating a model for scratch cards in Django:

```
from django.db import models from django.contrib.auth.models
import User class
ScratchCard(models.Model):
   user = models.ForeignKey(User, on_delete=models.CASCADE)
card_type = models.CharField(max_length=50)     card_value =
models.IntegerField()     card_pin =
models.CharField(max_length=50)     is_used =
models.BooleanField(default=False)     created_at =
models.DateTimeField(auto_now_add=True)     updated_at =
models.DateTimeField(auto_now=True)
   def __str__(self):          return
f"{self.card_type} ({self.card_value})"
```

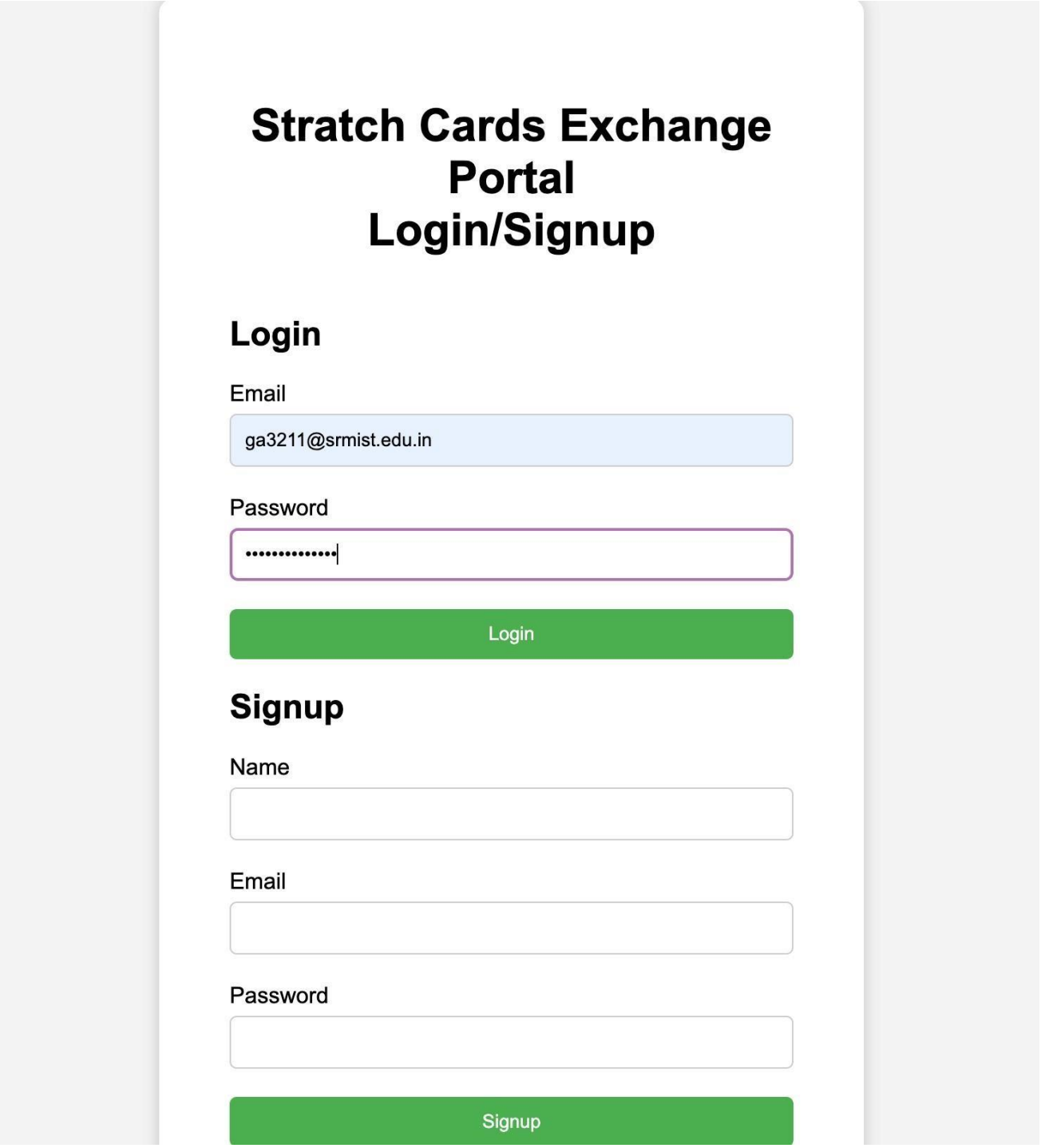
JavaScript code for handling user authentication and API requests:

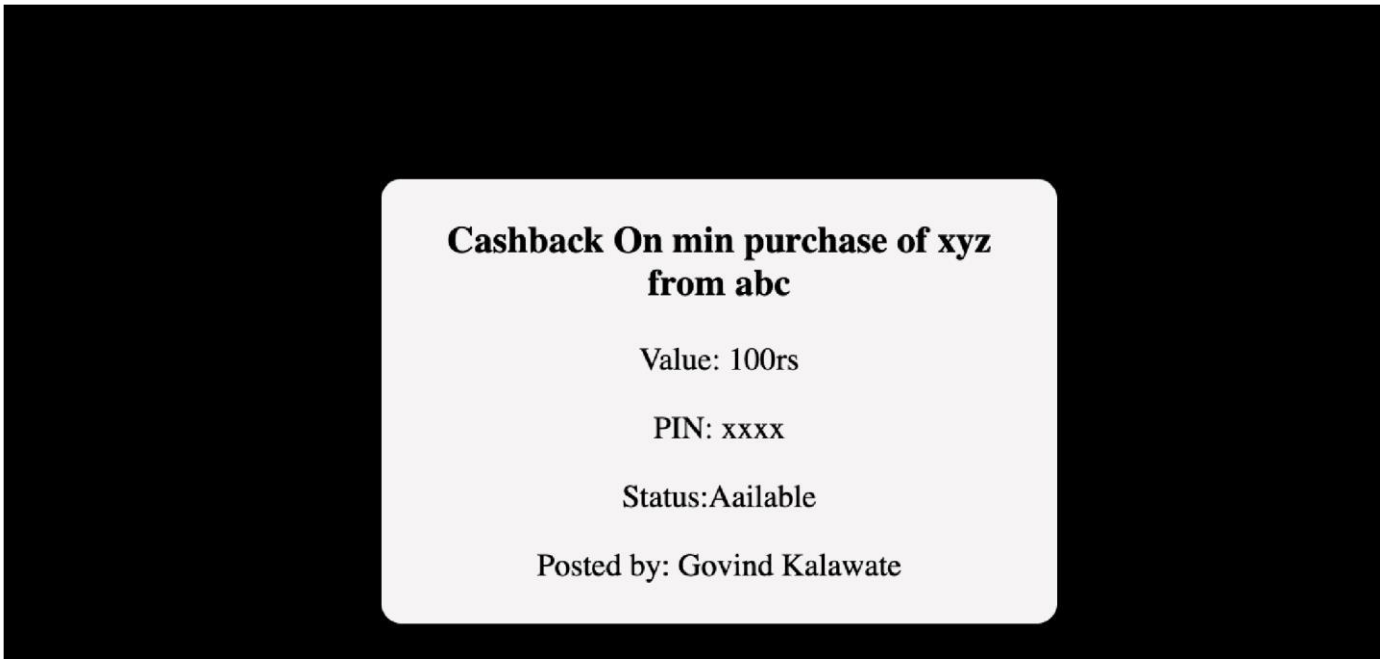
```
// User Authentication const loginForm = document.querySelector("#login-
form");

loginForm.addEventListener("submit", async (e) => {     e.preventDefault();
   const email = document.querySelector("#email").value;     const
password = document.querySelector("#password").value;     const
response = await fetch("/api/auth/token/login/", {         method:
"POST",         headers: {
          "Content-Type": "application/json",
       },
       body: JSON.stringify({ email, password }),
   });      if (!response.ok) {          const
error = await response.json();
       console.log(error);
   } else {          const data = await
response.json();
       localStorage.setItem("authToken", data.auth_token);
window.location.href = "/";
   }  });

// API requests
const getScratchCards = async () => {
   const response = await fetch("/api/scratch-cards/");
const data = await response.json();     return data;
};

const createScratchCard = async (card) => {     const
response = await fetch("/api/scratch-cards/", {         method:
"POST",         headers: {
          "Content-Type": "application/json",
          Authorization: `Token ${localStorage.getItem("authToken")}`,
       },
       body: JSON.stringify(card),
   });
   const data = await response.json();
return data; };
```

HTML and CSS code for displaying a list of scratch cards:

```
<div class="container">
  <h1>Scratch Cards</h1>
  <div class="card-list">
    {% for card in scratch_cards %}
    <div class="card">
      <h3>{{ card.card_type }}</h3>
      <p>Value: {{ card.card_value }}</p>
      <p>PIN: {{ card.card_pin }}</p>
      <p>Status: {% if card.is_used %}Used{% else %}Available{% endif %}</p>
      <p>Posted by: {{ card.user }}</p>
    </div>
    {% endfor %}
  </div> </div>

<style> .container {
max-width: 960px;
margin: 0 auto;   padding:
50px;
} h1 {    text-align:
center; }
```

SCREENSHOTS AND RESULTS





Success!

You have successfully exchanged your scratch card with another user. Thank you for using our platform.

Go back to Scratch Cards

My Cards

Uploaded for exchange/sell:

Boat 1000 rs off

Value:200 rs
PIN:xxxxx
Status: Available
Posted on: 06/05/2023
Valid Upto: 25/07/2023
Edit

Purchased/Exchanged:

Cash Back on min purchase of xyz from abc

Value: 300 rs
PIN: xxxxx
Exchanged on: 06/05/2023
Exchanged with: Govind

CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, we have created a web application for trading and buying scratch cards. Users can examine scratch cards published by other users, upload scratch cards for sale or exchange, and request exchanges or purchases using the application.

Future improvements might include the implementation of a payment mechanism to speed up user-to-user transactions. In order to make it simpler for customers to find particular kinds of scratch cards, we may also integrate a search option. To enhance security and stop unauthorised access to the programme, we may also integrate user authentication.

Overall, the application offers a practical exchange and sale platform for scratch cards and has the potential to be improved upon in the future.

REFERENCES

- Django documentation: https://docs.djangoproject.com/en/3.2/
- Bootstrap documentation: https://getbootstrap.com/docs/5.1/getting-started/introduction/
- Python documentation: https://docs.python.org/3/
- W3Schools: https://www.w3schools.com/
- Stack Overflow: https://stackoverflow.com/
- Medium: https://medium.com/
- Real Python: https://realpython.com/
- Django for Beginners book by William S. Vincent: https://djangoforbeginners.com/
- Python Crash Course book by Eric Matthes: https://ehmatthes.github.io/pcc_2e/
- Bootstrap 5 Quick Start book by Jacob Lett: https://www.apress.com/us/book/9781484276105