STUDENT DETAILS

REGISTRATION NO: COM/B/01-00162/2021

STUDENT NAME: MICHAEL OTIENO KASUKU

PROJECT TITLE

Bursary Mashinani: The Proposed Bursary Application Portal for Kisumu West Constituency

ABSTRACT

This document covers:

PART A: FEASIBILITY STUDY

1. **DEVELOPMENT ENVIRONMENT**
2. **PROPOSED IDEA**
3. **SYSTEM ARCHITECTURE**

PART B: REQUIREMENTS ENGINEERING

4. **SYSTEM REQUIREMENTS**
5. **SYSTEM MODELING**

PART C: HUMAN COMPUTER INTERACTION

6. **USER INTERFACE DESIGN**

PART D: IMPLEMENTATION

7. **DESIGN AND ANALYSIS OF ALGORITHMS**
8. **DEPLOYMENT**

1.DEVELOPMENT ENVIRONMENT

1.1 IDE: VS Codium

1.2 Programming language: Python

1.3 Web Framework: Django

1.4 CASE Tools: StarUML and Pencil Project

1.5 DBMS: PostgreSQL

1.6 Version Control: Git and Github

1.7 Deployment Environment: PythonAnywhere

1.1 VS CODIUM

VS Codium is essentially a branded version of Microsoft's Visual Studio Code (VS Code) that is provided without the telemetry tracking and licensing restrictions that are present in the official Visual Studio Code builds.

1.2 PYTHON

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991.

1.3 DJANGO

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.

1.3.1 KEY FEATURES OF DJANGO

- Object-Relational Mapping (ORM)
- URL routing
- Template engine
- Security features

1.3.2 WELL KNOWN APPS THAT USES DJANGO

- Instagram
- Bitbucket
- Pinterest

1.4.1 Pencil Project

Pencil Project is an open-source GUI prototyping tool that allows designers and developers to create mockups, wireframes, and prototypes for desktop, mobile, and web applications.

1.4.2 StarUML

StarUML is a sophisticated software modeling tool that supports various modeling languages such as Unified Modeling Language (UML) and Entity-Relationship Diagrams (ERD).

1.5 PostgreSQL

PostgreSQL is an object-relational database management system (ORDBMS) based on POSTGRES,Version 4.2 , developed at the University of California at Berkeley Computer Science Department.

1.5.1 Advantages of PostgreSQL

- Reliability
- Performance
- Scalability

1.5.2 Popular Apps Using PostgreSQL

- Instagram
- Netflix
- Uber
- Spotify

1.6.1 Git

Git is a distributed version control system (DVCS) used for tracking changes in source code during software development. It was created by Linus Torvalds in 2005 to manage the development of the Linux kernel, and it has since become one of the most widely used version control systems in the world.

1.6.1.1 Key features of Git

- Version Control
- Distributed
- Branching and merging
- Remote repositories
- Staging area
- Open source

1.6.2 Github

GitHub is a web-based platform built on top of Git, providing additional features and functionalities for hosting Git repositories and facilitating collaboration among developers. It is one of the most popular platforms for hosting Git repositories and managing software development projects.

1.6.2.1 Key features of Github

- Repository hosting
- Collaboration tools
- Forks and branches
- Github pages

1.7 PythonAnywhere

PythonAnywhere is an online integrated development environment (IDE) and web hosting service based on the Python programming language. It provides a platform for developers to write, run, and host Python applications entirely in the cloud, without requiring any installations on their local machines.

2. THE PROPOSED IDEA

2.1 Bursary Application Portal

2.2 Popular bursary application portals in Kenya

2.3 Why use Django to build bursary application portal

2.4 Bursary Mashinani

2.1 Bursary Application Portal

A **bursary application portal** is an online platform designed to facilitate the process of applying for bursaries or scholarships.

2.2 Popular Bursary Application Portals in Kenya

- The Higher Education Loans Board (HELB) Portal
- KCB Foundation Scholarships Portal
- The Equity Group Foundation Scholarships Portal

2.3 Why use Django to build Bursary Application Portal

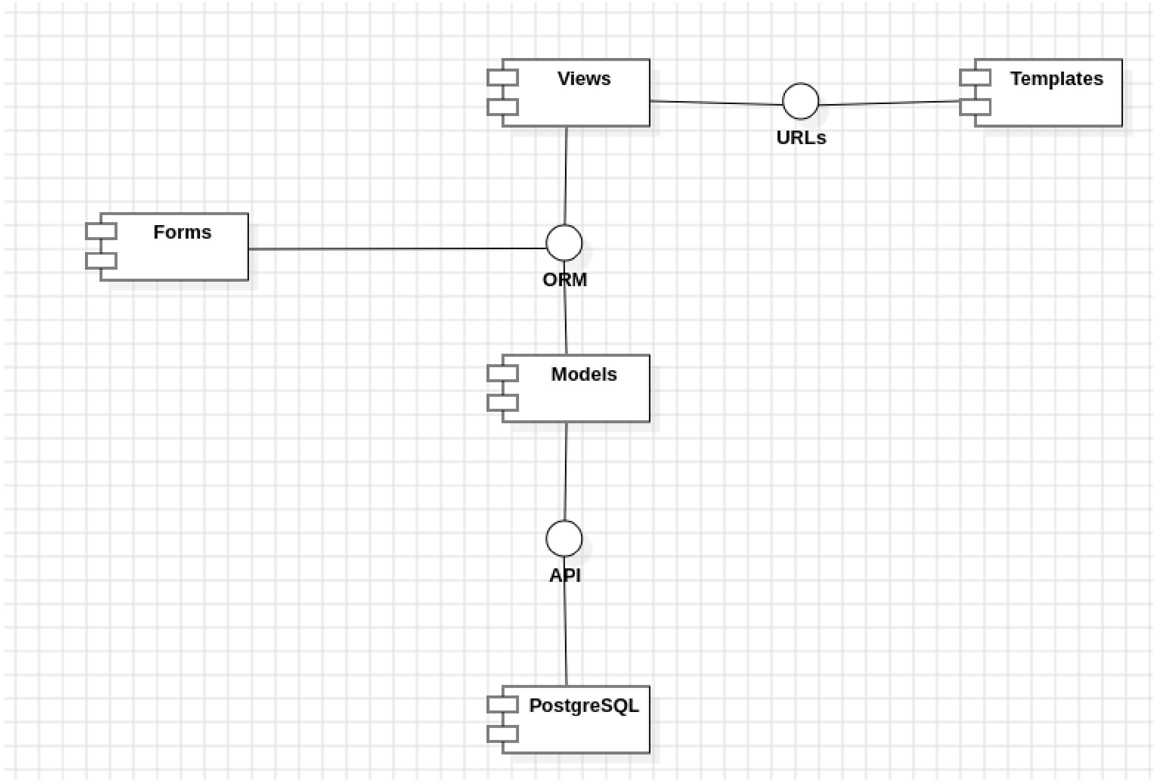
- Rapid Development
- Scalability
- Security

2.4 Bursary Mashinani

"Bursary Mashinani" is an innovative Bursary Application Portal designed to streamline the bursary application process for Kisumu West Constituency. This digital solution offers:

- Efficient Application Process
- Secure and User-Friendly Interface
- Track Progress Functionality

3. SYSTEM ARCHITECTURE



4. SYSTEM REQUIREMENTS

4.1 Functional Requirements

4.2 Non Functional Requirements

4.1 Functional Requirements

REQ-1: The portal shall allow the applicant to create a new user account.

REQ-2: The portal shall allow the applicant to login to the respective user account.

REQ -3: The portal shall ensure that the applicant can only apply for bursary once in every financial year.

REQ-4: The portal shall ensure that the national id number provided by the applicant belongs to that particular applicant based on the applicant’s registration number.

4.1 Functional Requirements

REQ-5: The portal shall ensure that the registration number provided by the applicant is actually a valid registration number based on the provided student register of the chosen institution.

REQ-6: The portal shall ensure that the applicant is indeed a resident of the chosen ward.

REQ-7: The portal shall ensure that the provided account number is indeed a valid account number of the chosen institution.

REQ-8: The portal shall ensure that the bursary application can be submitted only if the financial year status is open.

REQ-9: The portal shall ensure that the provided serial number is a valid serial number for the bursary application report to be generated.

4.1 Functional Requirements

REQ-10: The portal shall allow the admin to manage bank data.

REQ-11: The portal shall allow the admin to manage institution data.

REQ-12: The portal shall allow the admin to manage account data.

REQ -13: The portal shall allow the admin to manage country data.

REQ-14: The portal shall allow the admin to manage region data.

REQ-17: The portal shall allow the admin to manage county data.

REQ-18: The portal shall allow the admin to manage constituency data.

REQ-19: The portal shall allow the admin to manage ward data.

4.1 Functional Requirements

REQ-18: The portal shall allow the admin to manage resident data.

REQ-19: The portal shall allow the admin to manage student data.

REQ-20: The portal shall allow the admin to manage financial year data.

REQ-21: The portal shall allow the admin to manage bursary application data.

REQ-22: The portal shall allow the admin to manage user data.

REQ-23: The portal shall allow the admin to manage password reset token data.

4.2 Non Functional Requirements

4.2.1 Performance

4.2.1.1 Optimize database queries by ensuring that only necessary fields are retrieved from the database.

4.2.1.2 Optimize template rendering by minimizing the use of template tags and expressions, and by avoiding heavy computations or complex logic in templates.

4.2.1.3 Optimize URL patterns to avoid unnecessary redirects and route requests efficiently to the appropriate views, minimizing the number of requests and response times.

4.2 Non Functional Requirements

4.2.2 Scalability

4.2.2.1 Design scalable database schema that can accommodate increased data volume without performance degradation.

4.2.2.2 Design views to be stateless and independent of each other to facilitate horizontal scaling.

4.2.2.3 Use scalable URL patterns that are easy to manage and maintain, avoiding patterns that could lead to bottlenecks or resource contention as the application grows.

4.2 Non Functional Requirements

4.2.3 Reliability

4.2.3.1 Ensure data integrity by enforcing database constraints such as unique constraints, foreign key constraints, and NOT NULL constraints.

4.2.3.2 Implement error handling and logging in views to capture and handle exceptions gracefully, preventing application crashes and downtime.

4.2.3.3 Implement form validation to ensure that only valid data is submitted to the server, reducing the likelihood of data corruption or application errors.

4.2 Non Functional Requirements

4.2.4 Security

4.2.4.1 Protect sensitive data by encrypting it before storing it in the database.

4.2.4.2 Implement measures to prevent Cross-Site Request Forgery (CSRF) attacks, ensuring the security and integrity of user interactions with the application.

4.2.4.3 Use HTTPS to encrypt data transmitted between the client and server, preventing eavesdropping and tampering with sensitive information. Configure Django to enforce HTTPS by setting the SECURE_SSL_REDIRECT setting to True.

4.2 Non Functional Requirements

4.2.5 Usability

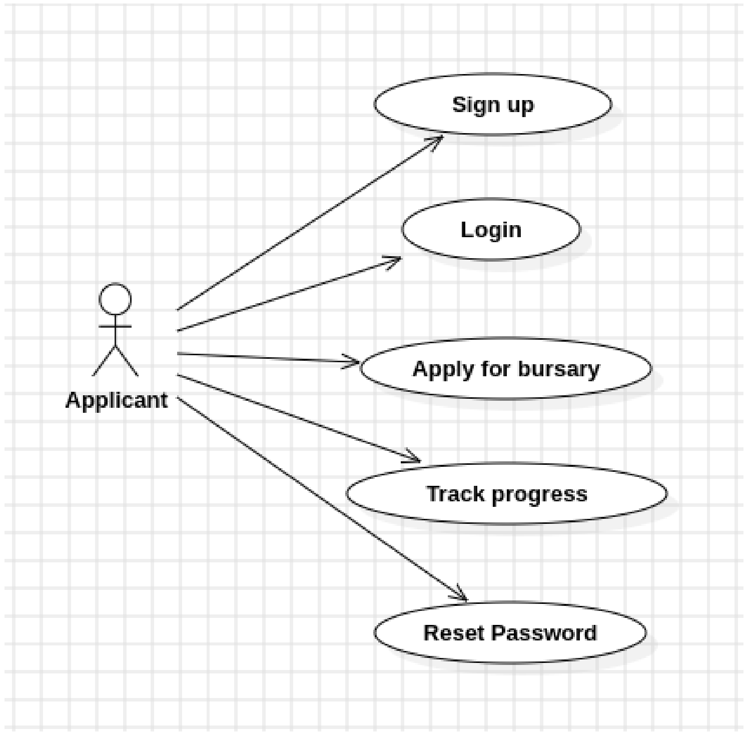
4.2.5.1 Design intuitive and user-friendly data models that accurately represent the real-world entities and relationships in the application domain. Use meaningful field names and relationships to make it easier for developers and users to understand the data model.

4.2.5.2 Design views to provide clear and informative feedback to users, guiding them through the application process and helping them understand their actions. Use descriptive error messages and success messages to provide feedback to users.

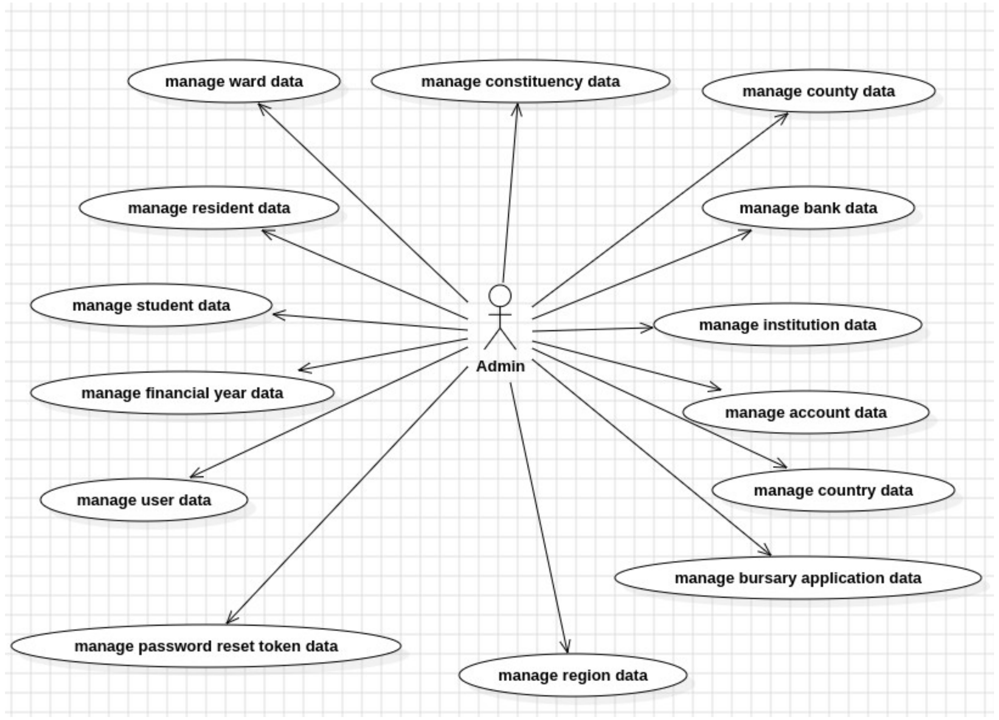
4.2.5.3 Design forms with clear labels, instructions, and error messages to help users complete them accurately and efficiently. Use Django's form widgets and labels to customize the appearance and behavior of form fields.

5. SYSTEM MODELING

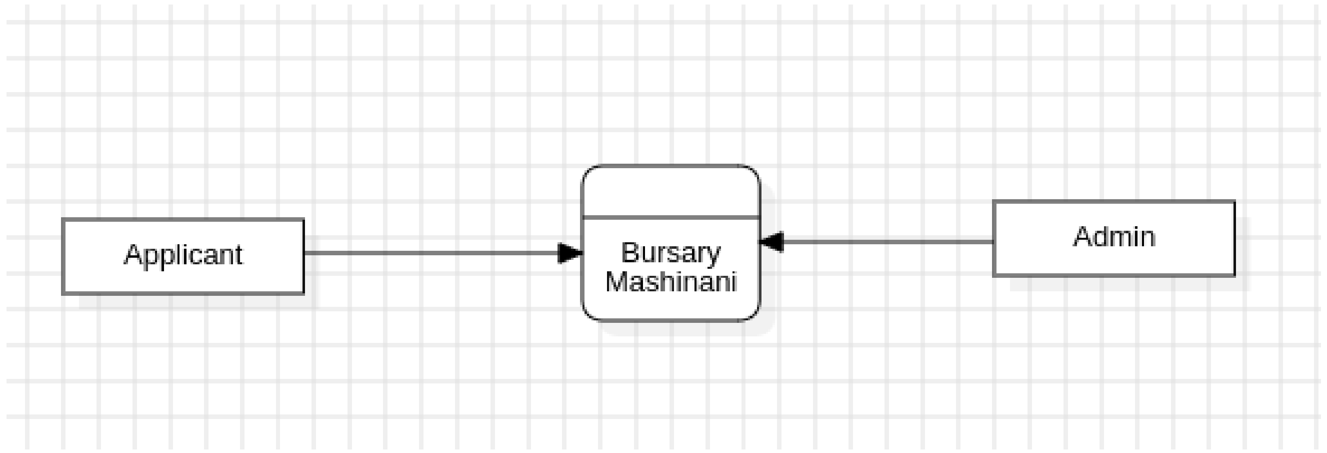
5.1.1 Use Case: Applicant



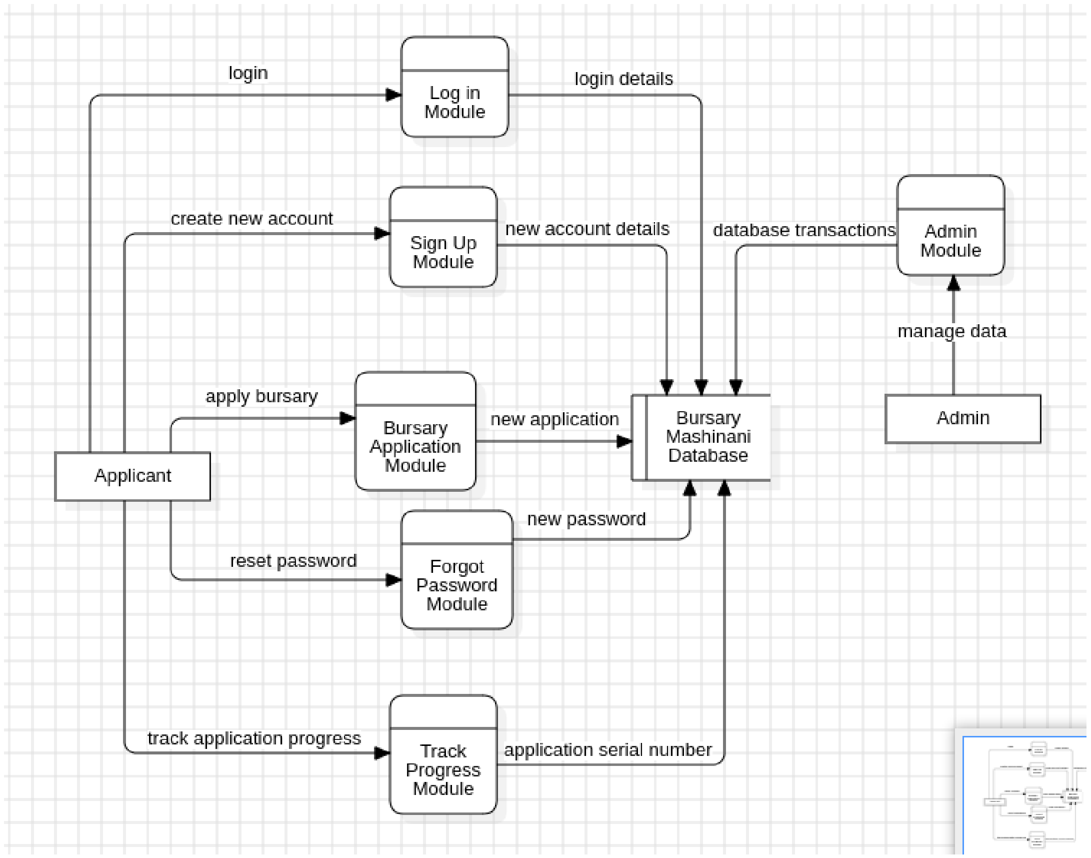
5.1.2 Use Case: Admin



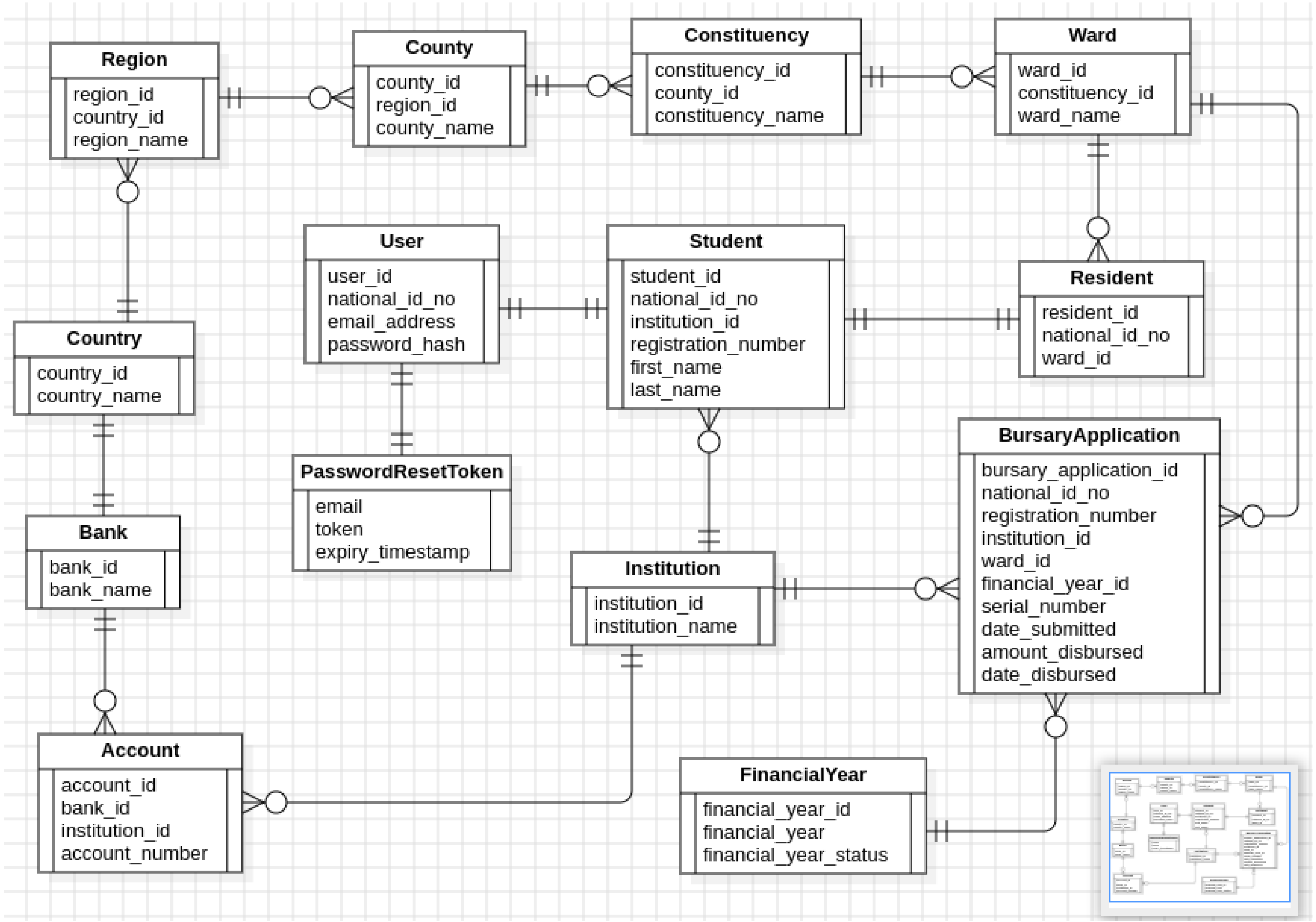
5.2 Context Diagram



5.3 Level 1 DFD

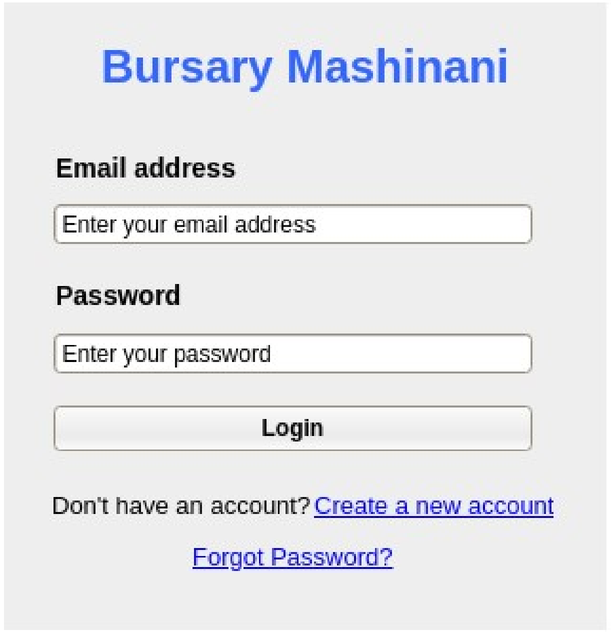


5.4 ER Diagram



6. USER INTERFACE DESIGN

6.1 Login Page


Bursary Mashinani

Email address

Enter your email address

Password

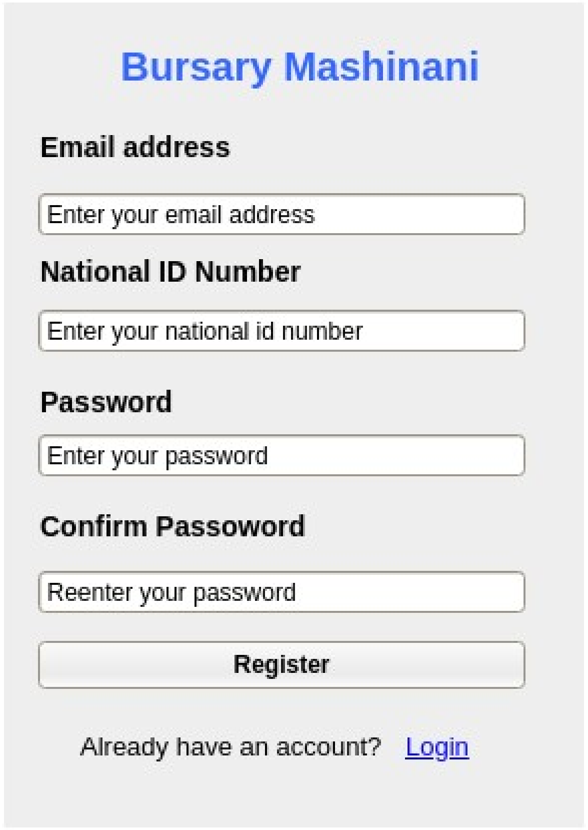
Enter your password

Login

Don't have an account? Create a new account

Forgot Password?

6.2 Sign Up Page



Bursary Mashinani

Email address

Enter your email address

National ID Number

Enter your national id number

Password

Enter your password

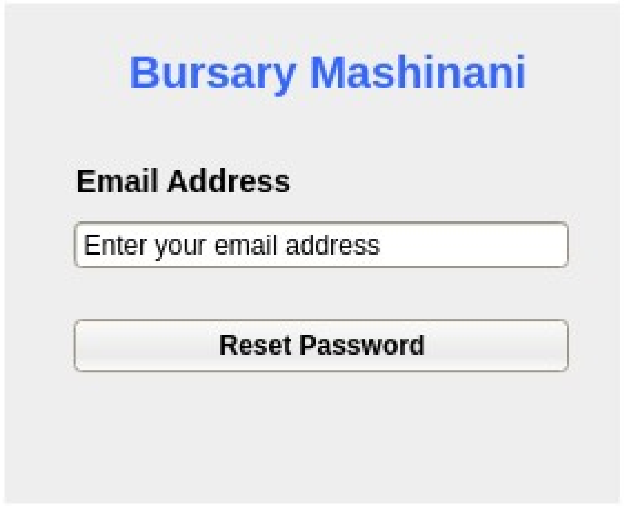
Confirm Passoword

Reenter your password

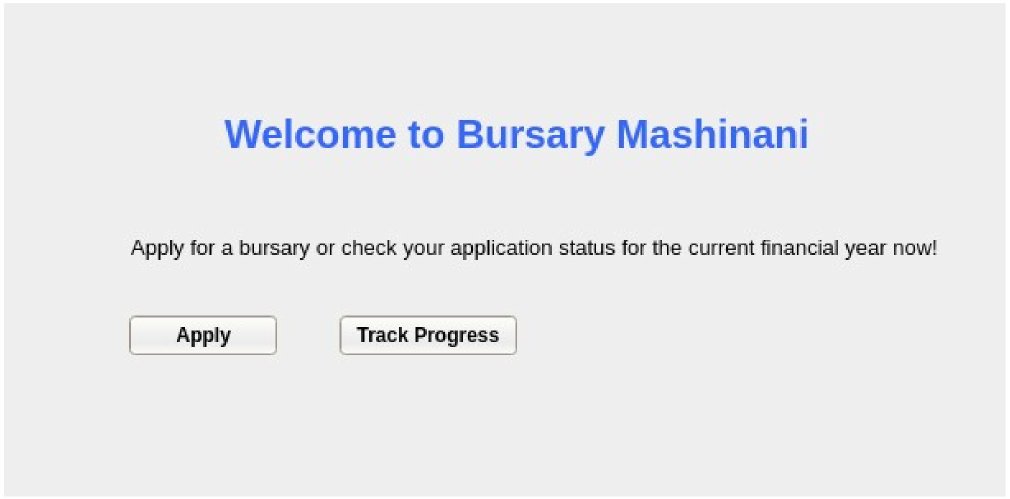
Register

Already have an account? Login

6.3 Forgot Password Page

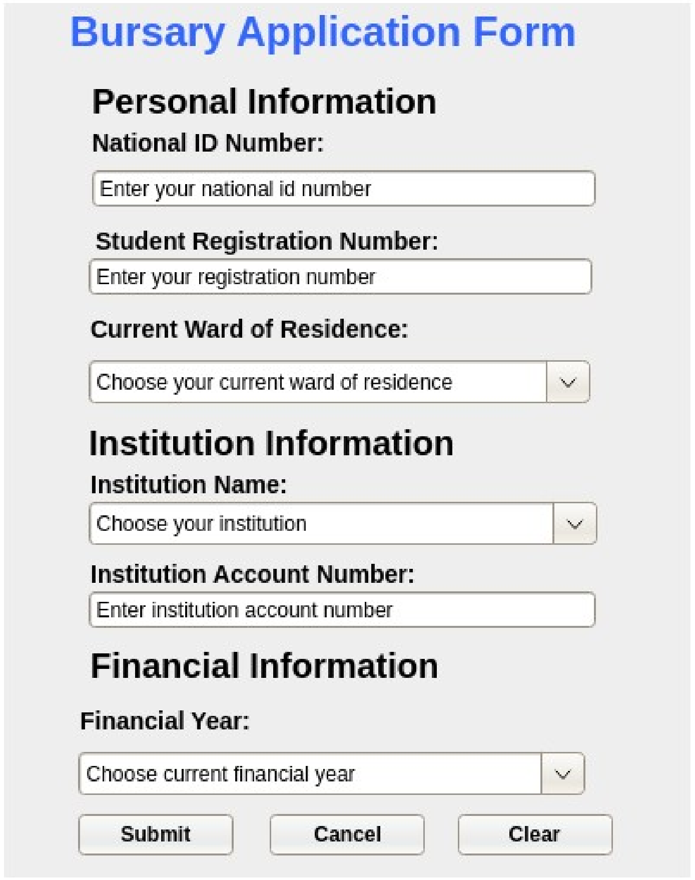


6.4 Landing Page


Welcome to Bursary Mashinani

Apply for a bursary or check your application status for the current financial year now!

6.5 Application Page



Bursary Application Form

Personal Information

National ID Number:

Enter your national id number

Student Registration Number:

Enter your registration number

Current Ward of Residence:

Choose your current ward of residence

Institution Information

Institution Name:

Choose your institution

Institution Account Number:

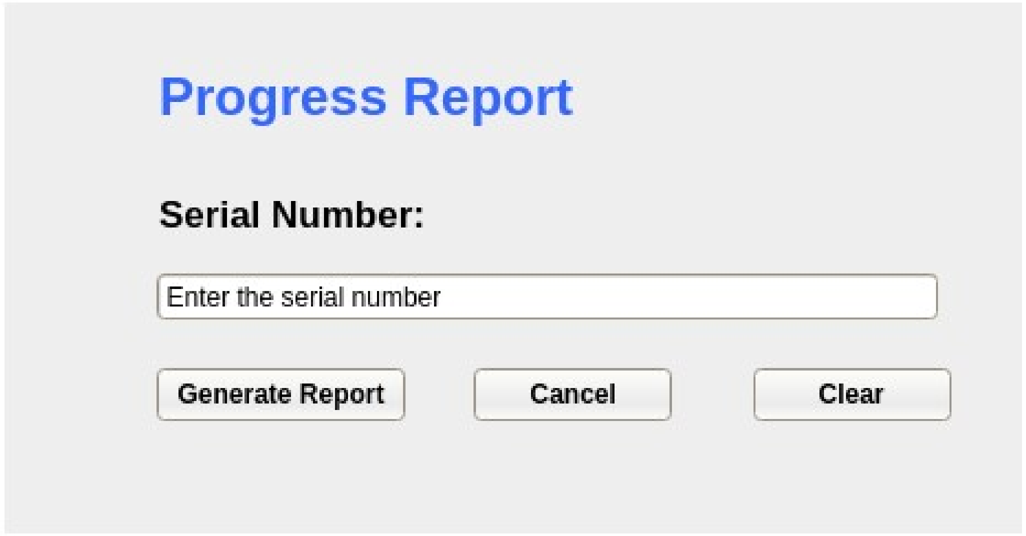
Enter institution account number

Financial Information

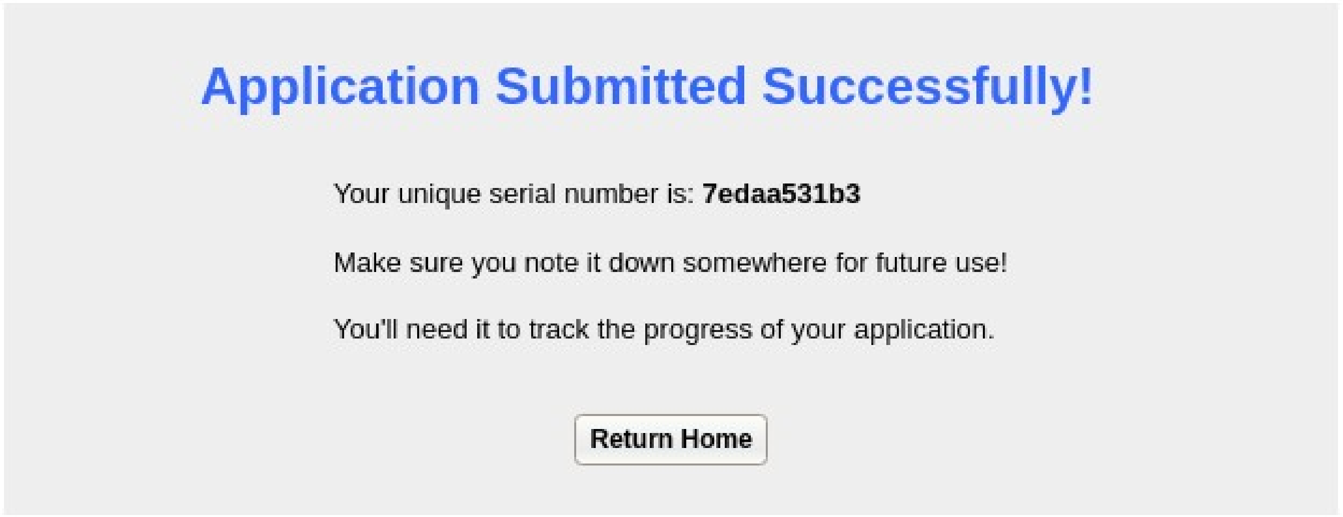
Financial Year:

Choose current financial year

6.6 Track Progress Page



6.7 Success Page

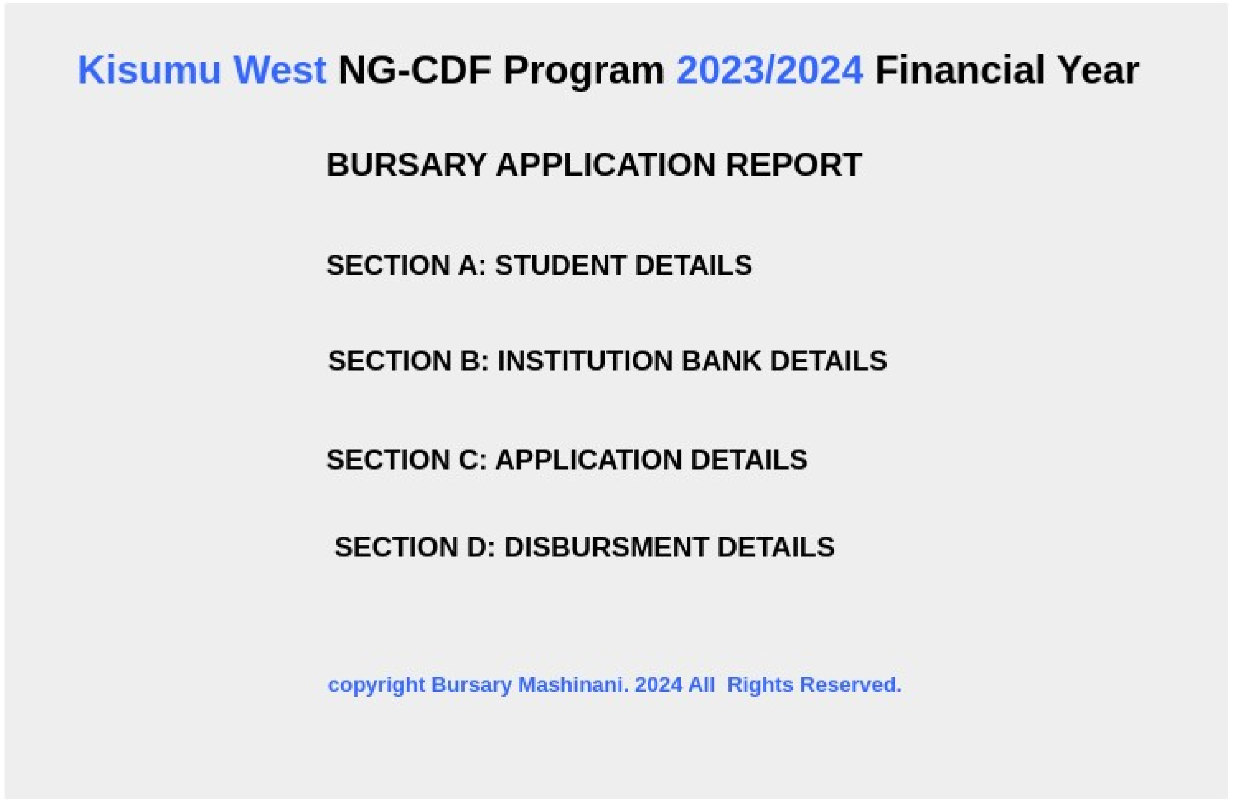

Application Submitted Successfully!

Your unique serial number is: **7edaa531b3**

Make sure you note it down somewhere for future use!

You'll need it to track the progress of your application.

6.8 Application Report Page



Kisumu West NG-CDF Program 2023/2024 Financial Year

BURSARY APPLICATION REPORT

SECTION A: STUDENT DETAILS

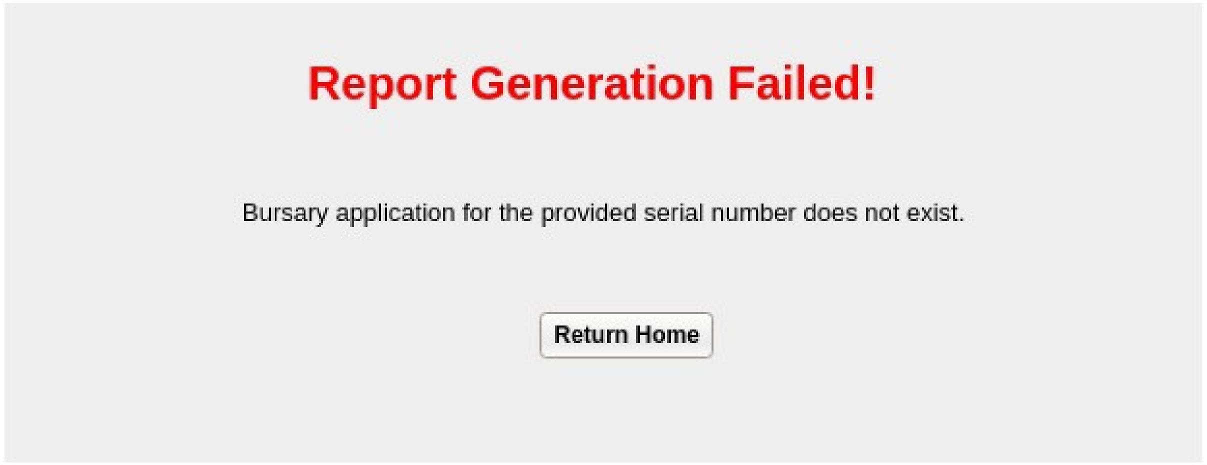
SECTION B: INSTITUTION BANK DETAILS

SECTION C: APPLICATION DETAILS

SECTION D: DISBURSMENT DETAILS

copyright Bursary Mashinani. 2024 All Rights Reserved.

6.9 Error Page


Report Generation Failed!

Bursary application for the provided serial number does not exist.

7. DESIGN AND ANALYSIS OF ALGORITHMS

7.1 Models

7.1.1 Bank

```
class Bank:
    // Attributes
    bank_id : Integer
    bank_name : String

    // Constructor
    Bank(bank_id, bank_name):
        set this.bank_id to bank_id
        set this.bank_name to bank_name

    // Method to return bank_name
    method __str__() returns String:
        return this.bank_name
```

7.1.2 Institution

```
class Institution:
    // Attributes
    institution_id : Integer
    institution_name : String

    // Constructor
    Institution(institution_id, institution_name):
        set this.institution_id to institution_id
        set this.institution_name to institution_name

    // Method to return institution_name
    method __str__() returns String:
        return this.institution_name
```

7.1.3 Account

```
class Account:
    // Attributes
    account_id : Integer
    institution_id : Integer // Foreign key to Institution
    bank_id : Integer // Foreign key to Bank
    account_number : String

    // Constructor
    Account(account_id, institution_id, bank_id, account_number):
        set this.account_id to account_id
        set this.institution_id to institution_id
        set this.bank_id to bank_id
        set this.account_number to account_number

    // Method to return account_number
    method __str__() returns String:
        return this.account_number
```

7.1.4 Country

```
class Country:
    // Attributes
    country_id : Integer
    country_name : String

    // Constructor
    Country(country_id, country_name):
        set this.country_id to country_id
        set this.country_name to country_name

    // Method to return country_name
    method __str__() returns String:
        return this.country_name
```

7.1.5 Region

```
class Region:
    // Attributes
    region_id : Integer
    country_id : Integer // Foreign key to Country
    region_name : String

    // Constructor
    Region(region_id, country_id, region_name):
        set this.region_id to region_id
        set this.country_id to country_id
        set this.region_name to region_name

    // Method to return region_name
    method __str__() returns String:
        return this.region_name
```

7.1.6 County

```
class County:
    // Attributes
    county_id : Integer
    region_id : Integer // Foreign key to Region
    county_name : String

    // Constructor
    County(county_id, region_id, county_name):
        set this.county_id to county_id
        set this.region_id to region_id
        set this.county_name to county_name

    // Method to return county_name
    method __str__() returns String:
        return this.county_name
```

7.1.7 Constituency

```
class Constituency:
    // Attributes
    constituency_id : Integer
    county_id : Integer // Foreign key to County
    constituency_name : String

    // Constructor
    Constituency(constituency_id, county_id, constituency_name):
        set this.constituency_id to constituency_id
        set this.county_id to county_id
        set this.constituency_name to constituency_name

    // Method to return constituency_name
    method __str__() returns String:
        return this.constituency_name
```

7.1.8 Ward

```
class Ward:
    // Attributes
    ward_id : Integer
    constituency_id : Integer // Foreign key to Constituency
    ward_name : String

    // Constructor
    Ward(ward_id, constituency_id, ward_name):
        set this.ward_id to ward_id
        set this.constituency_id to constituency_id
        set this.ward_name to ward_name

    // Method to return ward_name
    method __str__() returns String:
        return this.ward_name
```

7.1.9 Resident

```
class Resident:
    // Attributes
    resident_id : Integer
    national_id_no : String
    ward_id : Integer // Foreign key to Ward

    // Constructor
    Resident(resident_id, national_id_no, ward_id):
        set this.resident_id to resident_id
        set this.national_id_no to national_id_no
        set this.ward_id to ward_id

    // Method to return national_id_no
    method __str__() returns String:
        return this.national_id_no
```

7.1.10 Student

```
class Student:
    // Attributes
    student_id : Integer
    national_id_no : String
    institution_id : Integer // Foreign key to Institution
    registration_number : String
    first_name : String
    last_name : String

    // Constructor
    Student(student_id, national_id_no, institution_id, registration_number, first_name, last_name):
        set this.student_id to student_id
        set this.national_id_no to national_id_no
        set this.institution_id to institution_id
        set this.registration_number to registration_number
        set this.first_name to first_name
        set this.last_name to last_name

    // Method to return registration_number
    method __str__() returns String:
        return this.registration_number
```

7.1.11 Financial Year

```
class FinancialYear:
    // Attributes
    financial_year_id : Integer
    financial_year : String
    financial_year_status : String

    // Constructor
    FinancialYear(financial_year_id, financial_year, financial_year_status):
        set this.financial_year_id to financial_year_id
        set this.financial_year to financial_year
        set this.financial_year_status to financial_year_status

    // Method to return financial_year
    method __str__() returns String:
        return this.financial_year

// Validation function for financial year
function validate_financial_year(value) returns Void:
    parts = split value by '/'
    if length of parts is not 2:
        raise ValidationError('Invalid financial year!')

    AAAA, BBBB = parts[0], parts[1]
    if not (AAAA is a digit and BBBB is a digit):
        raise ValidationError('Invalid financial year!')

    AAAA, BBBB = convert AAAA to integer, convert BBBB to integer
    if not (1900 <= AAAA <= 2100 and 1900 <= BBBB <= 2100 and AAAA is equal to BBBB - 1):
        raise ValidationError('Invalid financial year!')

// Validation function for financial year status
function validate_financial_year_status(value) returns Void:
    if value not in ['Open', 'Closed']:
        raise ValidationError('Financial year status should be either "Open" or "Closed"!')
```

7.1.12 Bursary Application

```
class BursaryApplication:
    // Attributes
    bursary_application_id : Integer
    national_id_no : String
    registration_number : String
    institution_id : Integer // Foreign key to Institution
    account_number : String
    ward_id : Integer // Foreign key to Ward
    financial_year_id : Integer // Foreign key to FinancialYear
    serial_number : String
    date_submitted : DateTime
    amount_disbursed : Decimal
    date_disbursed : DateTime

    // Constructor
    BursaryApplication(bursary_application_id, national_id_no, registration_number, institution_id, account_number, ward_id, financial_year_id, serial_number, date_submitted, amount_disbursed, date_disbursed):
        set this.bursary_application_id to bursary_application_id
        set this.national_id_no to national_id_no
        set this.registration_number to registration_number
        set this.institution_id to institution_id
        set this.account_number to account_number
        set this.ward_id to ward_id
        set this.financial_year_id to financial_year_id
        set this.serial_number to serial_number
        set this.date_submitted to date_submitted
        set this.amount_disbursed to amount_disbursed
        set this.date_disbursed to date_disbursed

    // Validation function for serial number
    function validate_serial_number(value) returns Void:
        if not matches value against regex pattern r'^[a-zA-Z0-9]{10}$':
            raise ValidationError('Serial number must be exactly 10 alphanumeric characters!')

    // Clean method
    method clean() returns Void:
        super().clean()
        if this.date_disbursed and this.date_disbursed < this.date_submitted:
            raise ValidationError("Date disbursed should be greater than or equal to the date submitted!")

    // Method to return serial_number
    method __str__() returns String:
        return this.serial_number
```

7.1.13 User

```
class User:
    // Attributes
    user_id : Integer
    national_id_no : String
    email_address : String
    password_hash : String

    // Constructor
    User(user_id, national_id_no, email_address, password_hash):
        set this.user_id to user_id
        set this.national_id_no to national_id_no
        set this.email_address to email_address
        set this.password_hash to password_hash

    // Method to set password
    method set_password(raw_password) returns Void:
        this.password_hash = make_password(raw_password)

    // Method to check password
    method check_password(raw_password) returns Boolean:
        return check_password(raw_password, this.password_hash)

    // Custom validation for password field
    method clean() returns Void:
        if length of this.password_hash < 8:
            raise ValidationError("Password must be at least 8 characters long!")

    // Method to return email_address
    method __str__() returns String:
        return this.email_address
```

7.1.14 Password Reset Token

```
class PasswordResetToken:
    // Attributes
    email : String
    token : String
    expiry_timestamp : DateTime

    // Constructor
    PasswordResetToken(email, token, expiry_timestamp):
        set this.email to email
        set this.token to token
        set this.expiry_timestamp to expiry_timestamp

    // Method to return email
    method get_email() returns String:
        return this.email

    // Method to return token
    method get_token() returns String:
        return this.token

    // Method to return expiry_timestamp
    method get_expiry_timestamp() returns DateTime:
        return this.expiry_timestamp
```

7.2 Migrations

7.2.1 insert banks

7.2.2 insert institutions

7.2.3 insert account

7.2.4 insert countries

7.2.5 insert regions

7.2.6 insert counties

7.2.7 insert constituency

7.2.8 insert ward

7.2.9 insert residents

7.2.10 insert financial year

7.2.11 insert students

7.2.1 Insert banks

```
# Import necessary modules
import openpyxl

# Define a function to insert Kenyan banks into the database
function insert_kenyan_banks(apps, schema_editor):
    # Get the Bank model from the 'mashinani' app
    Bank = apps.get_model('mashinani', 'Bank')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Bank']

            # Extract bank names from the Excel sheet
            banks_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                bank_name = row[0]  # Assuming bank name is in the first column
                banks_data.append({'bank_name': bank_name})

            # Insert data into the model
            for data in banks_data:
                try:
                    Bank.objects.create(**data)
                except ValidationError as e:
                    print("Validation Error:", e)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0001_initial'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_kenyan_banks)
    ]
```

7.2.2 Insert institutions

```
# Import necessary modules
import openpyxl

# Define a function to insert Kenyan institutions into the database
function insert_kenyan_institutions(apps, schema_editor):
    # Get the Institution model from the 'mashinani' app
    Institution = apps.get_model('mashinani', 'Institution')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Institution']

            # Extract institution names from the Excel sheet
            institutions_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                institution_name = row[0]  # Assuming institution name is in the first column
                institutions_data.append({'institution_name': institution_name})

            # Insert data into the model
            for data in institutions_data:
                try:
                    Institution.objects.create(**data)
                except ValidationError as e:
                    print("Validation Error:", e)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0002_auto_20240403_1335'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_kenyan_institutions)
    ]
```

7.2.3 Insert account

```
# Import necessary modules
import openpyxl

# Define a function to insert account data into the database
function insert_account_data(apps, schema_editor):
    # Get models from the 'mashinani' app
    Account = apps.get_model('mashinani', 'Account')
    Institution = apps.get_model('mashinani', 'Institution')
    Bank = apps.get_model('mashinani', 'Bank')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Account']

            # Iterate over rows in the Excel sheet and extract account information
            accounts_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                institution_name = row[0]  # Assuming institution name is in the first column
                bank_name = row[1]  # Assuming bank name is in the second column
                account_number = row[2]  # Assuming account number is in the third column

                # Fetch institution and bank objects from the database
                try:
                    institution = Institution.objects.get(institution_name=institution_name)
                    bank = Bank.objects.get(bank_name=bank_name)
                except ObjectDoesNotExist as e:
                    print("Error:", e)
                    continue  # Skip this row if institution or bank doesn't exist

                # Create dictionary for account data
                account_data = {
                    'institution_id': institution,
                    'bank_id': bank,
                    'account_number': account_number
                }

                # Append account data to list
                accounts_data.append(account_data)

            # Insert data into the model
            for data in accounts_data:
                Account.objects.create(**data)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0003_auto_20240403_1338'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_account_data)
    ]
```

7.2.4 Insert countries

```
# Import necessary modules
import openpyxl

# Define a function to insert countries into the database
function insert_countries(apps, schema_editor):
    # Get the Country model from the 'mashinani' app
    Country = apps.get_model('mashinani', 'Country')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Country']

            # Extract country names from the Excel sheet
            countries_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                country_name = row[0]  # Assuming country name is in the first column
                countries_data.append({'country_name': country_name})

            # Insert data into the model
            for data in countries_data:
                try:
                    Country.objects.create(**data)
                except ValidationError as e:
                    print("Validation Error:", e)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0004_auto_20240403_1340'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_countries)
    ]
```

7.2.5 Insert regions

```
# Import necessary modules
import openpyxl

# Define a function to insert regions into the database
function insert_regions(apps, schema_editor):
    # Get models from the 'mashinani' app
    Region = apps.get_model('mashinani', 'Region')
    Country = apps.get_model('mashinani', 'Country')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Region']

            # Iterate over rows in the Excel sheet and extract region information
            regions_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                country_name = row[0]  # Assuming country name is in the first column
                region_name = row[1]  # Assuming region name is in the second column

                # Fetch country objects from the database
                try:
                    country = Country.objects.get(country_name=country_name)
                except ObjectDoesNotExist as e:
                    print("Error:", e)
                    continue  # Skip this row if country doesn't exist

                # Create dictionary for region data
                region_data = {
                    'country_id': country,
                    'region_name': region_name
                }

                # Append region data to list
                regions_data.append(region_data)

            # Insert data into the model
            for data in regions_data:
                Region.objects.create(**data)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0005_auto_20240403_1435'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_regions)
    ]
```

7.2.6 Insert counties

```
# Import necessary modules
import openpyxl

# Define a function to insert counties into the database
function insert_counties(apps, schema_editor):
    # Get models from the 'mashinani' app
    County = apps.get_model('mashinani', 'County')
    Region = apps.get_model('mashinani', 'Region')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['County']

            # Iterate over rows in the Excel sheet and extract county information
            counties_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                region_name = row[0]  # Assuming region name is in the first column
                county_name = row[1]  # Assuming county name is in the second column

                # Fetch region objects from the database
                try:
                    region = Region.objects.get(region_name=region_name)
                except ObjectDoesNotExist as e:
                    print("Error:", e)
                    continue  # Skip this row if region doesn't exist

                # Create dictionary for county data
                county_data = {
                    'region_id': region,
                    'county_name': county_name
                }

                # Append county data to list
                counties_data.append(county_data)

            # Insert data into the model
            for data in counties_data:
                County.objects.create(**data)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0006_auto_20240403_1437'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_counties)
    ]
```

7.2.7 Insert constituencies

```
# Define a function to insert constituencies into the database
function insert_constituencies(apps, schema_editor):
    # Get models from the 'mashinani' app
    Constituency = apps.get_model('mashinani', 'Constituency')
    County = apps.get_model('mashinani', 'County')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Constituency']

            # Iterate over rows in the Excel sheet and extract constituency information
            constituencies_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                county_name = row[0]  # Assuming county name is in the first column
                constituency_name = row[1]  # Assuming constituency name is in the second column

                # Fetch county objects from the database
                try:
                    county = County.objects.get(county_name=county_name)
                except ObjectDoesNotExist as e:
                    print("Error:", e)
                    continue  # Skip this row if county doesn't exist

                # Create dictionary for constituency data
                constituency_data = {
                    'county_id': county,
                    'constituency_name': constituency_name
                }

                # Append constituency data to list
                constituencies_data.append(constituency_data)

            # Insert data into the model
            for data in constituencies_data:
                Constituency.objects.create(**data)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0007_auto_20240403_1450'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_constituencies)
    ]
```

7.2.8 Insert wards

```
# Define a function to insert wards into the database
function insert_wards(apps, schema_editor):
    # Get models from the 'mashinani' app
    Ward = apps.get_model('mashinani', 'Ward')
    Constituency = apps.get_model('mashinani', 'Constituency')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Ward']

            # Iterate over rows in the Excel sheet and extract ward information
            wards_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                constituency_name = row[0]  # Assuming constituency name is in the first column
                ward_name = row[1]  # Assuming ward name is in the second column

                # Fetch constituency objects from the database
                try:
                    constituency = Constituency.objects.get(constituency_name=constituency_name)
                except ObjectDoesNotExist as e:
                    print("Error:", e)
                    continue  # Skip this row if constituency doesn't exist

                # Create dictionary for ward data
                ward_data = {
                    'constituency_id': constituency,
                    'ward_name': ward_name
                }

                # Append ward data to list
                wards_data.append(ward_data)

            # Insert data into the model
            for data in wards_data:
                Ward.objects.create(**data)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0008_auto_20240403_1515'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_wards)
    ]
```

7.2.9 Insert residents

```
# Define a function to insert residents into the database
function insert_residents(apps, schema_editor):
    # Get models from the 'mashinani' app
    Resident = apps.get_model('mashinani', 'Resident')
    Ward = apps.get_model('mashinani', 'Ward')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Resident']

            # Iterate over rows in the Excel sheet and extract resident information
            residents_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                national_id_no = row[0]  # Assuming National ID Number is in the first column
                ward_name = row[1]  # Assuming ward name is in the second column

                # Fetch ward objects from the database
                try:
                    ward = Ward.objects.get(ward_name=ward_name)
                except ObjectDoesNotExist as e:
                    print("Error:", e)
                    continue  # Skip this row if ward doesn't exist

                # Create dictionary for resident data
                resident_data = {
                    'national_id_no': national_id_no,
                    'ward_id': ward
                }

                # Append resident data to list
                residents_data.append(resident_data)

            # Insert data into the model
            for data in residents_data:
                Resident.objects.create(**data)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0009_auto_20240403_1519'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_residents)
    ]
```

7.2.10 Insert financial year

```
# Define a function to insert financial years into the database
function insert_financial_years(apps, schema_editor):
    # Get the FinancialYear model from the 'mashinani' app
    FinancialYear = apps.get_model('mashinani', 'FinancialYear')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['FinancialYear']

            # Iterate over rows in the Excel sheet and extract financial year information
            financial_years_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                financial_year = row[0]  # Assuming Financial Year is in the first column
                financial_year_status = row[1]  # Assuming Financial Year Status is in the second column

                # Create dictionary for financial year data
                financial_year_data = {
                    'financial_year': financial_year,
                    'financial_year_status': financial_year_status
                }

                # Append financial year data to list
                financial_years_data.append(financial_year_data)

            # Insert data into the model
            for data in financial_years_data:
                FinancialYear.objects.create(**data)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0010_auto_20240403_1612'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_financial_years)
    ]
```

7.2.11 Insert students

```
# Define a function to insert students into the database
function insert_students(apps, schema_editor):
    # Get models from the 'mashinani' app
    Student = apps.get_model('mashinani', 'Student')
    Institution = apps.get_model('mashinani', 'Institution')

    try:
        # Attempt to load data from an Excel workbook
        try:
            workbook = openpyxl.load_workbook('mashinani/data/data.xlsx')
            sheet = workbook['Student']

            # Iterate over rows in the Excel sheet and extract student information
            students_data = []
            for row in sheet.iter_rows(min_row=2, values_only=True):  # Assuming data starts from second row
                national_id_no = row[0]  # Assuming National ID Number is in the first column
                institution_name = row[1]
                registration_number = row[2]
                first_name = row[3]
                last_name = row[4]

                # Fetch institution objects from the database
                try:
                    institution = Institution.objects.get(institution_name=institution_name)
                except ObjectDoesNotExist as e:
                    print("Error:", e)
                    continue  # Skip this row if institution doesn't exist

                # Create dictionary for student data
                student_data = {
                    'national_id_no': national_id_no,
                    'institution_id': institution,
                    'registration_number': registration_number,
                    'first_name': first_name,
                    'last_name': last_name,
                }

                # Append student data to list
                students_data.append(student_data)

            # Insert data into the model
            for data in students_data:
                Student.objects.create(**data)

        # Handle the case where the file is not found
        except FileNotFoundError:
            print("File not found. Please check the path to the Excel file.")

        # Handle any other exceptions
        except Exception as e:
            print("An error occurred:", e)

    # Handle any exceptions
    except Exception as e:
        print("An error occurred:", e)

# Define a migration class
class Migration(migrations.Migration):
    # Define dependencies
    dependencies = [
        ('mashinani', '0011_auto_20240403_1618'),
    ]

    # Define operations
    operations = [
        migrations.RunPython(insert_students)
    ]
```

7.3 Forms

7.3.1 Register Form

7.3.2 Login Form

7.3.3 Password Reset Form

7.3.4 Application Form

7.3.1 Register Form

```
class LoginForm extends Form:
    email_address = EmailField(
        label="Email Address",
        widget=EmailInput(attrs={'class': 'form-control', 'placeholder': 'Enter your email address'})
    )
    password = CharField(
        label="Password",
        widget=PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Enter your password'})
    )

    def clean():
        cleaned_data = super().clean()
        email_address = cleaned_data.get("email_address")
        password = cleaned_data.get("password")
        return cleaned_data
```

7.3.2 Login Form

```
class LoginForm extends Form:
    email_address = EmailField(
        label="Email Address",
        widget=EmailInput(attrs={'class': 'form-control', 'placeholder': 'Enter your email address'})
    )
    password = CharField(
        label="Password",
        widget=PasswordInput(attrs={'class': 'form-control', 'placeholder': 'Enter your password'})
    )

    def clean():
        cleaned_data = super().clean()
        email_address = cleaned_data.get("email_address")
        password = cleaned_data.get("password")
        return cleaned_data
```

7.3.3 PasswordReset Form

```
class PasswordResetForm extends Form:
    email_address = EmailField(
        label='Email Address',
        widget=EmailInput(attrs={'class': 'form-control', 'placeholder': 'Enter your email address'})
    )

    def clean_email_address():
        email = self.cleaned_data.get('email_address')
        if not User.objects.filter(email_address=email).exists():
            raise ValidationError("This email address is not associated with any account!")
        return email
```

7.3.4 Application Form

```
class ApplicationForm extends ModelForm:
    institution_id = ModelChoiceField(
        queryset=Institution.objects.all(),
        required=True,
        label='Institution Name',
        widget=Select(attrs={'class': 'blue-input-box', 'placeholder': 'Select an institution'}),
    )
    financial_year_id = ModelChoiceField(
        queryset=FinancialYear.objects.filter(financial_year_status='Open'),
        required=True,
        label='Financial Year',
        widget=Select(attrs={'class': 'blue-input-box', 'placeholder': 'Select a financial year'}),
    )
    ward_id = ModelChoiceField(
        queryset=Ward.objects.all(),
        required=True,
        label='Current Ward of Residence',
        widget=Select(attrs={'class': 'blue-input-box', 'placeholder': 'Select a ward'}),
    )
    Meta:
        model = BursaryApplication
        fields = ['national_id_no', 'registration_number', 'institution_id', 'account_number', 'ward_id', 'financial_year_id']
        labels = {
            'national_id_no': 'National ID Number',
            'registration_number': 'Student Registration Number',
            'institution_id': 'Institution Name',
            'account_number': 'Institution Account Number',
            'ward_id': 'Current Ward of Residence',
            'financial_year_id': 'Financial Year',
        }
        widgets = {
            'national_id_no': TextInput(attrs={'class': 'blue-input-box', 'placeholder': 'Enter national ID number'}),
            'registration_number': TextInput(attrs={'class': 'blue-input-box', 'placeholder': 'Enter registration number'}),
            'account_number': TextInput(attrs={'class': 'blue-input-box', 'placeholder': 'Enter institution account number'}),
        }

    def __init__():
        super().__init__()

        # Correctly set the initial value based on the existing instance
        if self.instance and hasattr(self.instance, 'institution_id') and self.instance.institution_id:
            self.fields['institution_id'].initial = self.instance.institution.institution_id
```

7.4 Views

7.4.1 Login View

7.4.2 Register View

7.4.3 Register Success Page View

7.4.4 Password Reset View

7.4.5 Password Reset Success View

7.4.6 Landing Page View

7.4.7 Application Form View

7.4.8 Success Page View

7.4.9 Progress Report View

7.4.10 generate pdf

7.4.1 Login View

```
class LoginView extends View:
    template_name = 'login.html'

    def get(request):
        form = LoginForm()
        return render(request, template_name, {'form': form})

    def post(request):
        form = LoginForm(request.POST)
        if form.is_valid():
            email_address = form.cleaned_data['email_address']
            password = form.cleaned_data['password']
            user = User.objects.filter(email_address=email_address).first()
            if user and user.check_password(password):
                # Authentication successful
                request.session['email_address'] = user.email_address
                return redirect('landing_page')
            else:
                # Authentication failed
                form.add_error(None, "Incorrect email address or password!")
                return render(request, template_name, {'form': form}, status=400)
        else:
            return render(request, template_name, {'form': form}, status=400)
```

7.4.2 Register View

```
class RegisterView extends View:
    template_name = 'signup.html'

    def get(request):
        form = RegisterForm()
        return render(request, template_name, {'form': form})

    def post(request):
        form = RegisterForm(request.POST)

        if form.is_valid():
            national_id_no = form.cleaned_data['national_id_no']
            email_address = form.cleaned_data['email_address']
            password_hash = form.cleaned_data['password_hash']

            # Check if account already exists
            if User.objects.filter(email_address=email_address).exists():
                form.add_error(None, "This email address has already been used!")
                return render(request, template_name, {'form': form})
            else:
                # Create the account
                new_account = form.save(commit=False)
                new_account.set_password(password_hash)
                new_account.save()

                # Redirect to success page
                return redirect('register_success')
        else:
            # If the form is not valid, render the template with the form and errors
            return render(request, template_name, {'form': form})
```

7.4.3 Register Success Page View

```
class RegisterSuccessPageView extends View:
    def get(request):
        return render(request,
'register_success.html')
```

7.4.4 Password Reset View

```
class PasswordResetView extends View:
    template_name = 'forgot_password.html'

    def get(request):
        form = PasswordResetForm()
        return render(request, template_name, {'form': form})

    def post(request):
        form = PasswordResetForm(request.POST)

        if form.is_valid():
            email_address = form.cleaned_data['email_address']

            # REQ-1: Check if the email address exists
            if not User.objects.filter(email_address=email_address).exists():
                form.add_error(None, "Invalid email address!")
            else:
                # Generate a unique token
                token = generate_random_string(length=32)

                # Store the token along with the user's email address and expiration timestamp
                PasswordResetToken.objects.create(
                    email=email_address,
                    token=token,
                    expiry_timestamp=now() + timedelta(hours=1)  # Token expires in 1 hour
                )

                # Construct the password reset link
                reset_link = build_absolute_uri(request, f'/password-reset/{token}')

                # Send email with the reset link
                send_mail(
                    'Password Reset Link',
                    f'Click the following link to reset your password: {reset_link}',
                    'michaelotienokasuku@gmail.com',
                    [email_address],
                    fail_silently=False,
                )

                # Redirect to a success page or display a success message
                return redirect('password_reset_success', email_address=email_address)

        # If there are errors, render the template with the form and errors
        return render(request, template_name, {'form': form})
```

7.4.5 Password Reset Success View

```
class PasswordResetSuccessView extends View:
    def get(request, *args, **kwargs):
        email_address = self.kwargs.get('email_address', None)
        return render(request, 'password_reset_success.html', {'email_address': email_address})
```

7.4.6 Landing Page View

```
class LandingPageView extends View:
    def get(request):
        return render(request, 'landing_page.html')
```

7.4.7 Application Form View

```
class ApplicationFormView extends View:
    template_name = 'application_form.html'

    def get(request):
        form = ApplicationForm()
        return render(request, template_name, {'form': form})

    def post(request):
        form = ApplicationForm(request.POST)

        if form.is_valid():
            # Form is valid, proceed with processing data
            national_id_no = form.cleaned_data['national_id_no']
            registration_number = form.cleaned_data['registration_number']
            ward_id = form.cleaned_data['ward_id']
            institution_id = form.cleaned_data['institution_id']
            account_number = form.cleaned_data['account_number']
            financial_year_id = form.cleaned_data['financial_year_id']

            # Check for existing application based on the id number, registration number, and financial year
            if BursaryApplication.objects.filter(national_id_no=national_id_no, registration_number=registration_number, financial_year_id=financial_year_id).exists():
                form.add_error(None, "You have already applied for bursary for this financial year!")

            # Check if the id number provided belongs to that student
            elif not Student.objects.filter(national_id_no=national_id_no, registration_number=registration_number).exists():
                form.add_error(None, "You have provided a wrong registration number or national id number!")

            # Check student registration, i.e., if the applicant is a student of the given institution
            elif not Student.objects.filter(institution_id=institution_id, registration_number=registration_number).exists():
                form.add_error(None, "You have chosen the wrong institution or provided a wrong registration number!")

            # Check if the student is indeed a resident of the chosen ward based on the national id number and the chosen ward
            elif not Resident.objects.filter(national_id_no=national_id_no, ward_id=ward_id).exists():
                form.add_error(None, "You have entered a wrong national id number or chosen the wrong ward")

            # Check if the provided account number is correct based on the chosen institution
            elif not Account.objects.filter(institution_id=institution_id, account_number=account_number).exists():
                form.add_error(None, "You have entered a wrong account number or chosen the wrong institution")
            else:
                # Generate serial number
                serial_number = generate_serial_number(national_id_no, registration_number, financial_year_id, institution_id)

                # Save the application
                bursary_application = form.save(commit=False)
                bursary_application.serial_number = serial_number
                bursary_application.save()

                return redirect('success_page', serial_number=serial_number)

        # If there are errors, render the template with the form and errors
        return render(request, template_name, {'form': form})
```

Generate serial number

```
function generate_serial_number(national_id_no, registration_number, financial_year_id, institution_id):
    data_string = concatenate(national_id_no, "-", registration_number, "-", financial_year_id, "-", institution_id)
    unique_identifier = generate_uuid()
    combined_string = concatenate(data_string, "-", unique_identifier)
    hashed_serial = hash_sha256(combined_string)
    truncated_hash = get_substring(hashed_serial, 0, 10)
    return truncated_hash
```

7.4.8 Success Page View

```
class SuccessPageView extends View:
    def get(request, *args, **kwargs):
        serial_number =
self.kwargs.get('serial_number', None)
        return render(request, 'success_page.html',
{'serial_number': serial_number})
```

7.4.9 Progress Report View

```
class ProgressReportView extends View:
    def get(request):
        return render(request, 'progress_report.html')

    def post(request):
        serial_number = request.POST.get('serial_number')
        try:
            bursary_application = BursaryApplication.objects.get(serial_number=serial_number)
            student = Student.objects.get(registration_number=bursary_application.registration_number)
            account = Account.objects.get(account_number=bursary_application.account_number)
            ward = Ward.objects.get(ward_name=bursary_application.ward_id)
            constituency = ward.constituency_id
            county = constituency.county_id
        except BursaryApplication.DoesNotExist:
            return render(request, 'error_page.html')

        report_data = {
            'student_details': {
                'first_name': student.first_name,
                'last_name': student.last_name,
                'national_id_no': bursary_application.national_id_no,
                'registration_number': bursary_application.registration_number,
                'institution_id': bursary_application.institution_id,
                'ward_id': bursary_application.ward_id,
                'constituency_name': constituency.constituency_name,
                'county_name': county.county_name,
            },
            'account_details': {
                'bank_name': account.bank_id,
                'account_number': bursary_application.account_number,
            },
            'application_details': {
                'serial_number': bursary_application.serial_number,
                'financial_year_id': bursary_application.financial_year_id,
                'date_submitted': bursary_application.date_submitted,
            },
            'disbursement_details': {
                'amount_disbursed': bursary_application.amount_disbursed,
                'date_disbursed': bursary_application.date_disbursed,
            },
        }

        # Generate PDF
        pdf_bytes = generate_pdf(report_data)

        # Return the PDF file as a response
        response = HttpResponse(pdf_bytes, content_type='application/pdf')
        response['Content-Disposition'] = f'attachment; filename="{serial_number}_report.pdf"'
        return response
```

7.4.10 Generate pdf

```
function generate_pdf(report_data):
    buffer = create_bytes_io_buffer()
    pdf_canvas = create_pdf_canvas(buffer, pagesize=landscape(letter))

    # Define styles
    styles = get_sample_styles()
    title_style = styles['Heading1']
    title_style.text_color = colors.black  # Black font color
    title_style.alignment = 1  # Center alignment
    title_style.font_size = 28

    subtitle_style = styles['Heading2']
    subtitle_style.text_color = colors.black  # Black font color
    subtitle_style.alignment = 1  # Center alignment
    subtitle_style.font_size = 24

    section_header_style = styles['Heading3']
    section_header_style.text_color = colors.black  # Black font color
    section_header_style.font_size = 20

    # Define paragraph styles
    paragraph_style = create_paragraph_style(
        "Normal",
        font_size=18,
        leading=18,
        text_color="black",
        alignment=0  # 0 for left alignment, 1 for center, 2 for right
    )

    footer_style = styles['Normal']
    footer_style.alignment = 1  # Center alignment

    # Define footer content
    footer_text = "<font color='#01689b'>© 2024 Bursary Mashinani. All rights reserved.</font>"

    # Create footer paragraph
    footer = create_paragraph(footer_text, footer_style)

    # Create title
    title_text = concatenate("<font color='#01689b'>", report_data['student_details']['constituency_name'], "</font> NG-CDF <font color='#01689b'>", report_data['application_details']['financial_year_id'], "</font> Financial Year")
    title = create_paragraph(title_text, title_style)

    # Create subtitle
    subtitle_text = "Bursary Application Report"
    subtitle = create_paragraph(subtitle_text, subtitle_style)

    # Define sections
    sections = [
        ("<p>SECTION A: Student Details", [
            concatenate("<p><b>First Name:</b> ", report_data['student_details']['first_name'], "</p>"),
            concatenate("<p><b>Last Name:</b> ", report_data['student_details']['last_name'], "</p>"),
            concatenate("<p><b>National ID Number:</b> ", report_data['student_details']['national_id_no'], "</p>"),
            concatenate("<p><b>Registration Number:</b> ", report_data['student_details']['registration_number'], "</p>"),
            concatenate("<p><b>Institution Name:</b> ", report_data['student_details']['institution_id'], "</p>"),
            concatenate("<p><b>Current County of Residence:</b> ", report_data['student_details']['county_name'], "</p>"),
            concatenate("<p><b>Current Constituency of Residence:</b> ", report_data['student_details']['constituency_name'], "</p>"),
            concatenate("<p><b>Current Ward of Residence:</b> ", report_data['student_details']['ward_id'], "</p>")
        ]),
        ("<p>SECTION B: Institution Bank Details", [
            concatenate("<p><b>Bank Name:</b> ", report_data['account_details']['bank_name'], "</p>"),
            concatenate("<p><b>Institution Bank Account Number:</b> ", report_data['account_details']['account_number'], "</p>")
        ]),
        ("<p>SECTION C: Bursary Application Details", [
            concatenate("<p><b>Application Serial Number:</b> ", report_data['application_details']['serial_number'], "</p>"),
            concatenate("<p><b>Financial Year:</b> ", report_data['application_details']['financial_year_id'], "</p>"),
            concatenate("<p><b>Date Applied:</b> ", report_data['application_details']['date_submitted'], "</p>")
        ]),
        ("<p>SECTION D: Bursary Disbursement Details", [
            concatenate("<p><b>Amount Disbursed(Ksh.):</b> ", report_data['disbursement_details']['amount_disbursed'], "</p>"),
            concatenate("<p><b>Date Disbursed:</b> ", report_data['disbursement_details']['date_disbursed'], "</p>")
        ])
    ]

    # Create elements
    elements = [title, create_spacer(1, 0.25 * inch), subtitle, create_spacer(1, 0.5 * inch)]

    # Add sections to the PDF
    for section_title, section_content in sections:
        elements.append(create_paragraph(section_title, section_header_style))
        elements.extend([create_paragraph(content, paragraph_style) for content in section_content])
        elements.append(create_spacer(1, 0.25 * inch))

    elements.append(footer)

    # Build PDF
    build_pdf(pdf_canvas, elements)

    # Save PDF file
    pdf_bytes = get_buffer_value(buffer)
    close_buffer(buffer)

    return pdf_bytes
```

7.5 CSS Styles

7.5.1 Sign up

7.5.2 Log in

7.5.3 Forgot Password

7.5.4 Password Reset Success

7.5.5 Register Success

7.5.1 Sign up

```
// Define styles for register container
register-container:
  display: flex
  justify-content: center
  align-items: center
  height: 100vh

// Define styles for register form
register-form:
  max-width: 400px
  width: 100%
  padding: 40px
  background-color: #ffffff
  border-radius: 10px
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1)
  text-align: center

// Define styles for form title
form-title:
  text-align: center
  color: #3498db
  margin-bottom: 20px

// Define styles for form group
form-group:
  margin-bottom: 20px
  text-align: left

// Define styles for form group label
form-group label:
  font-weight: bold
  display: block
  margin-bottom: 5px

// Define styles for form control
form-control:
  width: calc(100% - 20px)
  padding: 10px
  font-size: 16px
  border: 1px solid #ccc
  border-radius: 5px

// Define styles for primary button
btn-primary:
  background-color: #007bff
  border: none
  color: #fff
  padding: 12px 20px
  font-size: 18px
  border-radius: 5px
  cursor: pointer    transition: background-color 0.3s ease

// Define styles for hover state of primary button
btn-primary:hover:
  background-color: #0056b3

// Define styles for error message
error-message:
  color: #ff0000
  font-size: 14px

// Define styles for text link
text-link:
  color: #007bff
  text-decoration: none

// Define styles for hover state of text link
text-link:hover:
  text-decoration: underline
```

7.5.2 Log in

```
// Define styles for container
container:
    display: flex
    justify-content: center
    align-items: center
    height: 100vh

// Define styles for login form
login-form:
    max-width: 350px
    width: 100%
    padding: 30px
    background-color: #fff
    border-radius: 10px
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1)

// Define styles for form title
form-title:
    text-align: center
    color: #3498db
    margin-bottom: 20px

// Define styles for form group
form-group:
    margin-bottom: 20px

// Define styles for form group label
form-group label:
    font-weight: bold
    display: block
    margin-bottom: 5px

// Define styles for form control
form-control:
    width: 100%
    padding: 10px
    font-size: 16px
    border: 1px solid #ccc
    border-radius: 5px

// Define styles for primary button
btn-primary:
    background-color: #007bff
    border: none
    color: #fff
    padding: 12px 20px
    font-size: 18px
    border-radius: 5px
    cursor: pointer
    transition: background-color 0.3s ease

// Define styles for hover state of primary button
btn-primary:hover:
    background-color: #0056b3

// Define styles for error message
error-message:
    color: #ff0000
    font-size: 14px

// Define styles for text center alignment
text-center:
    text-align: center

// Define styles for text link
text-link:
    color: #007bff
    text-decoration: none

// Define styles for hover state of text link
text-link:hover:
    text-decoration: underline
```

7.5.3 Forgot Password

```
FUNCTION applyStyles(element, styles):
    FOR EACH style IN styles:
        SET element.style[style.property] TO style.value

containerStyles = [
    { property: "margin-top", value: "50px" }
]

passwordResetFormStyles = [    { property: "max-width", value: "400px" },    { property: "margin", value: "0 auto" }]

cardStyles = [    { property: "border", value: "none" },    { property: "border-radius", value: "10px" },    { property: "box-shadow", value: "0 4px 8px rgba(0, 0, 0, 0.1)" }]

cardBodyStyles = [    { property: "padding", value: "2rem" }]

cardTitleStyles = [    { property: "font-size", value: "1.5rem" },    { property: "text-align", value: "center" },    { property: "color", value: "#3498db" }]

formGroupStyles = [    { property: "margin-bottom", value: "1.5rem" }]

formControlStyles = [    { property: "border", value: "1px solid #ccc" },    { property: "border-radius", value: "5px" },    { property: "padding", value: "0.5rem" },    { property: "font-size", value: "1rem" }]

btnPrimaryStyles = [    { property: "background-color", value: "#007bff" },    { property: "border", value: "none" },    { property: "border-radius", value: "5px" },    { property: "padding", value: "0.75rem" },    { property: "font-size", value: "1rem" },    { property: "cursor", value: "pointer" },    { property: "color", value: "#fff" },    { property: "transition", value: "background-color 0.3s" }]

btnPrimaryHoverStyles = [    { property: "background-color", value: "#0056b3" }]

errorMessageStyles = [    { property: "color", value: "#ff0000" },    { property: "font-size", value: "14px" }]

containerElement = SELECT_ELEMENT(".container")
applyStyles(containerElement, containerStyles)

passwordResetFormElement = SELECT_ELEMENT(".password-reset-form")
applyStyles(passwordResetFormElement, passwordResetFormStyles)

cardElements = SELECT_ALL_ELEMENTS(".card")
FOR EACH card IN cardElements:
    applyStyles(card, cardStyles)

cardBodyElements = SELECT_ALL_ELEMENTS(".card-body")
FOR EACH cardBody IN cardBodyElements:
    applyStyles(cardBody, cardBodyStyles)

cardTitleElements = SELECT_ALL_ELEMENTS(".card-title")
FOR EACH cardTitle IN cardTitleElements:
    applyStyles(cardTitle, cardTitleStyles)

formGroupElements = SELECT_ALL_ELEMENTS(".form-group")
FOR EACH formGroup IN formGroupElements:
    applyStyles(formGroup, formGroupStyles)

formControlElements = SELECT_ALL_ELEMENTS(".form-control")
FOR EACH formControl IN formControlElements:
    applyStyles(formControl, formControlStyles)

btnPrimaryElements = SELECT_ALL_ELEMENTS(".btn-primary")
FOR EACH btnPrimary IN btnPrimaryElements:
    applyStyles(btnPrimary, btnPrimaryStyles)

btnPrimaryHoverElements = SELECT_ALL_ELEMENTS(".btn-primary:hover")
FOR EACH btnPrimaryHover IN btnPrimaryHoverElements:
    applyStyles(btnPrimaryHover, btnPrimaryHoverStyles)

errorMessageElements = SELECT_ALL_ELEMENTS(".error-message")
FOR EACH errorMessage IN errorMessageElements:
    applyStyles(errorMessage, errorMessageStyles)
```

7.5.4 Password Reset Success

```
FUNCTION applyStyles(element, styles):
    FOR EACH style IN styles:
        SET element.style[style.property] TO style.value

successContainerStyles = [
    { property: "text-align", value: "center" },
    { property: "margin", value: "50px auto" },
    { property: "padding", value: "20px" },
    { property: "background-color", value: "#f0f8ff" },
    { property: "border-radius", value: "10px" },
    { property: "box-shadow", value: "0 0 10px rgba(0, 0, 0, 0.1)" }
]

successTitleStyles = [    { property: "color", value: "#0066cc" }]

textLinkStyles = [    { property: "color", value: "#007bff" },    { property: "text-decoration", value: "none" }]

textLinkHoverStyles = [    { property: "text-decoration", value: "underline" }]

successContainerElement = SELECT_ELEMENT(".success-container")
applyStyles(successContainerElement, successContainerStyles)

successTitleElement = SELECT_ELEMENT(".success-title")
applyStyles(successTitleElement, successTitleStyles)

textLinkElements = SELECT_ALL_ELEMENTS(".text-link")
FOR EACH textLink IN textLinkElements:
    applyStyles(textLink, textLinkStyles)

textLinkHoverElements = SELECT_ALL_ELEMENTS(".text-link:hover")
FOR EACH textLinkHover IN textLinkHoverElements:
    applyStyles(textLinkHover, textLinkHoverStyles)
```

7.5.5 Register Success

```
FUNCTION applyStyles(element, styles):
    FOR EACH style IN styles:
        SET element.style[style.property] TO style.value

successContainerStyles = [
    { property: "text-align", value: "center" },
    { property: "margin", value: "50px auto" },
    { property: "padding", value: "20px" },
    { property: "background-color", value: "#f0f8ff" },
    { property: "border-radius", value: "10px" },
    { property: "box-shadow", value: "0 0 10px rgba(0, 0, 0, 0.1)" }
]

successTitleStyles = [
    { property: "color", value: "#0066cc" }
]

textLinkStyles = [
    { property: "color", value: "#007bff" },
    { property: "text-decoration", value: "none" }
]

textLinkHoverStyles = [
    { property: "text-decoration", value: "underline" }
]

successContainerElements = SELECT_ALL_ELEMENTS(".success-container")
FOR EACH successContainer IN successContainerElements:
    applyStyles(successContainer, successContainerStyles)

successTitleElements = SELECT_ALL_ELEMENTS(".success-title")
FOR EACH successTitle IN successTitleElements:
    applyStyles(successTitle, successTitleStyles)

textLinkElements = SELECT_ALL_ELEMENTS(".text-link")
FOR EACH textLink IN textLinkElements:
    applyStyles(textLink, textLinkStyles)

textLinkHoverElements = SELECT_ALL_ELEMENTS(".text-link:hover")
FOR EACH textLinkHover IN textLinkHoverElements:
    applyStyles(textLinkHover, textLinkHoverStyles)
```

7.6 Scripts

7.6.1 Application form

7.6.2 Progress report

7.6.1 Application form

```
FUNCTION cancelForm():
    landingPageUrl = GET_ELEMENT_BY_ID('bursaryForm').getAttribute('data-landing-page-url')
    IF confirm("Are you sure you want to cancel? Any unsaved data will be lost."):
        SET window.location.href TO landingPageUrl

FUNCTION clearForm():
    IF confirm("Are you sure you want to clear the form? Any unsaved data will be lost."):
        GET_ELEMENT_BY_ID('bursaryForm').reset()
```

7.6.2 Progress report

```
FUNCTION cancelForm():
    landingPageUrl = GET_ELEMENT_BY_ID('reportForm').getAttribute('data-landing-page-url')
    IF confirm("Are you sure you want to cancel? Any unsaved data will be lost."):
        SET window.location.href TO landingPageUrl

FUNCTION clearForm():
    IF confirm("Are you sure you want to clear the form? Any unsaved data will be lost."):
        GET_ELEMENT_BY_ID('reportForm').reset()
```

7.7 URLs

```
DEFINE urlpatterns[]:
    path('/login/', LoginView, name='login')
    path('/register/', RegisterView, name='register')
    path('/password/reset/', PasswordResetView, name='password_reset')
    path('/home/', LandingPageView, name='landing_page')
    path('/apply/', ApplicationFormView, name='apply')
    path('/success/<str:serial_number>/', SuccessPageView, name='success_page')
    path('/progress_report/', ProgressReportView, name='progress_report')
    path('/password_reset_success/<str:email_address>/', PasswordResetSuccessView,
name='password_reset_success')
    path('/register/success/', RegisterSuccessPageView, name='register_success')
```

7.8 Templates

7.8.1 Base authentication

```
OUTPUT "{% load static %}"

OUTPUT "<!DOCTYPE html>"
OUTPUT "<html lang='en'>"
OUTPUT "<head>"
OUTPUT "    <meta charset='UTF-8'>"
OUTPUT "    <meta name='viewport' content='width=device-width, initial-scale=1.0'>"
OUTPUT "</head>"
OUTPUT "<body>"
OUTPUT "    {% block content %}{% endblock %}"
OUTPUT "</body>"
OUTPUT "</html>"
```

7.8.2 Base

```
OUTPUT "{% load static %}"

OUTPUT "<!DOCTYPE html>"
OUTPUT "<html lang='en'>"
OUTPUT "<head>"
OUTPUT "    <meta charset='UTF-8'>"
OUTPUT "    <meta name='viewport' content='width=device-width, initial-scale=1.0'>"
OUTPUT "    <title>Bursary Mashinani</title>"
OUTPUT "    <link rel='stylesheet' href='{% static 'css/main.css' %}'>"
OUTPUT "</head>"
OUTPUT "<body>"
OUTPUT "    <header class='header'>"
OUTPUT "        <h1 class='page-title-white'>Bursary Mashinani</h1>"
OUTPUT "        <nav>"
OUTPUT "            <a href='#' class='logout-button'>Logout</a>"
OUTPUT "        </nav>"
OUTPUT "    </header>"
OUTPUT "    <div class='container'>"
OUTPUT "        {% block content %}{% endblock %}"
OUTPUT "    </div>"
OUTPUT "    <footer class='footer'>"
OUTPUT "        <p>&copy; 2024 Bursary Mashinani. All rights reserved.</p>"
OUTPUT "    </footer>"
OUTPUT "    <script>"
OUTPUT "        document.querySelector('.logout-button').addEventListener('click', function(event) {"
OUTPUT "            event.preventDefault();"
OUTPUT "            var confirmation = confirm('Are you sure you want to logout?');"
OUTPUT "            if (confirmation) {"
OUTPUT "                window.location.href = '{% url 'login' %}';"
OUTPUT "            }"
OUTPUT "        });"
OUTPUT "    </script>"
OUTPUT "</body>"
OUTPUT "</html>"
```

7.8.3 Log in

```
OUTPUT "{% extends 'base_authentication.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<link rel='stylesheet' href='{% static 'css/login.css' %}'>"
OUTPUT "<div class='container'>"
OUTPUT "    <div class='login-form'>"
OUTPUT "        <h2 class='form-title'>Bursary Mashinani</h2>"
OUTPUT "        <form method='post' action='{% url 'login' %}' id='login-form'>"
OUTPUT "            {% csrf_token %}"
OUTPUT "            {% if form.errors %}"
OUTPUT "            <div class='error-message' role='alert'>"
OUTPUT "                {% for field, error_list in form.errors.items %}"
OUTPUT "                    {% for error in error_list %}"
OUTPUT "                        {{ error }}"
OUTPUT "                    {% endfor %}"
OUTPUT "                {% endfor %}"
OUTPUT "            </div>"
OUTPUT "            {% endif %}"
OUTPUT "            <div class='form-group'>"
OUTPUT "                <label for='id_email_address'>Email Address</label>"
OUTPUT "                {{ form.email_address }}"
OUTPUT "                <span id='email-error' class='error-message'>{{ form.errors.email_address }}</span>"
OUTPUT "            </div>"
OUTPUT "            <div class='form-group'>"
OUTPUT "                <label for='id_password'>Password</label>"
OUTPUT "                {{ form.password }}"
OUTPUT "                <span id='password-error' class='error-message'>{{ form.errors.password }}</span>"
OUTPUT "            </div>"
OUTPUT "            <button type='submit' class='btn-primary'>Login</button>"
OUTPUT "        </form>"
OUTPUT "        <div class='text-center'>"
OUTPUT "            <p>Don't have an account? <a href='{% url 'register' %}' class='text-link'>Register</a></p>"
OUTPUT "            <p><a href='{% url 'password_reset' %}' class='text-link'>Forgot your password?</a></p>"
OUTPUT "        </div>"
OUTPUT "    </div>"
OUTPUT "</div>"
OUTPUT "{% endblock %}"
```

7.8.4 Sign up

```
OUTPUT "{% extends 'base_authentication.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<link rel='stylesheet' href='{% static 'css/signup.css' %}'>"
OUTPUT "<div class='register-container'>"
OUTPUT "    <div class='register-form'>"
OUTPUT "        <h2 class='form-title'>Bursary Mashinani</h2>"
OUTPUT "        <form method='post' action='{% url 'register' %}' id='registration-form'>"
OUTPUT "            {% csrf_token %}"
OUTPUT "            {% if form.errors %}"
OUTPUT "            <div class='error-message' role='alert'>"
OUTPUT "                {% for field, error_list in form.errors.items %}"
OUTPUT "                    {% for error in error_list %}"
OUTPUT "                        {{ error }}"
OUTPUT "                    {% endfor %}"
OUTPUT "                {% endfor %}"
OUTPUT "            </div>"
OUTPUT "            {% endif %}"
OUTPUT "            {% for field in form %}"
OUTPUT "                <div class='form-group'>"
OUTPUT "                    <label for='{{ field.id_for_label }}'>{{ field.label }}</label>"
OUTPUT "                    {% if field.name == 'password_hash' %}"
OUTPUT "                        {{ field }}"
OUTPUT "                    {% else %}"
OUTPUT "                        {{ field }}"
OUTPUT "                    {% endif %}"
OUTPUT "                    {% if field.errors %}"
OUTPUT "                        {% for error in field.errors %}"
OUTPUT "                            <span class='error-message'>{{ error }}</span>"
OUTPUT "                            {% endfor %}"
OUTPUT "                    {% endif %}"
OUTPUT "                </div>"
OUTPUT "            {% endfor %}"
OUTPUT "            <button type='submit' class='btn-primary'>Sign Up</button>"
OUTPUT "        </form>"
OUTPUT "            <p>Already have an account? <a href='{% url 'login' %}' class='text-link'>Log In</a></p>"
OUTPUT "        </div>"
OUTPUT "    </div>"
OUTPUT "</div>"
OUTPUT "{% endblock %}"
```

7.8.5 Register success

```
OUTPUT "{% extends 'base_authentication.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<link rel='stylesheet' href='{% static 'css/register_success.css' %}'>"
OUTPUT "<div class='success-container'>"
OUTPUT "    <h2 class='success-title'>Account Created Successfully!</h2>"
OUTPUT "    <a href='{% url 'login' %}' class='text-link'>Back to Login</a>"
OUTPUT "</div>"
OUTPUT "{% endblock %}"
```

7.8.6 Forgot password

```
OUTPUT "{% extends 'base_authentication.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<link rel='stylesheet' href='{% static 'css/forgot_password.css' %}'>"
OUTPUT "<div class='container'>"
OUTPUT "    <div class='password-reset-form'>"
OUTPUT "        <div class='card'>"
OUTPUT "            <div class='card-body'>"
OUTPUT "                <h2 class='card-title'>Bursary Mashinani</h2>"
OUTPUT "                <form method='post' action='{% url 'password_reset' %}' id='password-reset-form'>"
OUTPUT "                    {% csrf_token %}"
OUTPUT "                    {% if form.errors %}"
OUTPUT "                    <div class='error-message' role='alert'>"
OUTPUT "                        {% for field, error_list in form.errors.items %}"
OUTPUT "                            {% for error in error_list %}"
OUTPUT "                                {{ error }}"
OUTPUT "                            {% endfor %}"
OUTPUT "                        {% endfor %}"
OUTPUT "                    </div>"
OUTPUT "                    {% endif %}"
OUTPUT "                    <div class='form-group'>"
OUTPUT "                        <label for='email'><b>Email address</b></label>"
OUTPUT "                        {{ form.email_address }}"
OUTPUT "                        <div id='error-message' class='error-message'>{{ form.errors.email_address }}</div>"
OUTPUT "                    </div>"
OUTPUT "                    <button type='submit' class='btn-primary'>Reset Password</button>"
OUTPUT "                </form>"
OUTPUT "            </div>"
OUTPUT "        </div>"
OUTPUT "    </div>"
OUTPUT "</div>"
OUTPUT "{% endblock %}"
```

7.8.7 Password Reset success

```
OUTPUT "{% extends 'base_authentication.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<link rel='stylesheet' href='{% static 'css/password_reset_success.css' %}'>"
OUTPUT "<div class='success-container'>"
OUTPUT "    <h2 class='success-title'>Email Sent Successfully!</h2>"
OUTPUT "    <p>Check: <b>{{ email_address|slice:':3' }}****{{ email_address|slice:'-4:' }}</b></p>"
OUTPUT "    <p>For more information on how to change your password!</p>"
OUTPUT "    <a href='{% url 'login' %}' class='text-link'>Back to Login</a>"
OUTPUT "</div>"
OUTPUT "{% endblock %}"
```

7.8.8 Landing Page

```
OUTPUT "{% extends 'base.html' %}"


OUTPUT "{% block content %}"
OUTPUT "<div class='container'>"
OUTPUT "    <h1 class='page-title'>Welcome to Bursary Mashinani</h1>"
OUTPUT "    <p class='intro-text'>Apply for a bursary or check your application status for the current financial year now!</p>"
OUTPUT "    <a href='{% url 'apply' %}' class='blue-button'>Apply</a>"
OUTPUT "    <a href='{% url 'progress_report' %}' class='green-button'>Track Progress</a>"
OUTPUT "</div>"
OUTPUT "{% endblock %}"
```

7.8.9 Application form

```
OUTPUT "{% extends 'base.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<div class='form-container'>"
OUTPUT "    <h1 class='page-title'>Bursary Application Form</h1>"
OUTPUT "    <form method='post' action='{% url 'apply' %}' id='bursaryForm' data-landing-page-url='{% url 'landing_page' %}'>"
OUTPUT "        {% csrf_token %}"
OUTPUT "        {% if form.errors %}"
OUTPUT "        <div class='error-message' role='alert'>"
OUTPUT "            {% for field, error_list in form.errors.items %}"
OUTPUT "                {% for error in error_list %}"
OUTPUT "                    {{ error }}"
OUTPUT "                {% endfor %}"
OUTPUT "            {% endfor %}"
OUTPUT "        </div>"
OUTPUT "        {% endif %}"
OUTPUT "        <div class='form-section'>"
OUTPUT "            <h2>Personal Information Section</h2>"
OUTPUT "            <div>"
OUTPUT "                <label for='{{ form.national_id_no.id_for_label }}' class='required'>National ID Number:</label>"
OUTPUT "                {{ form.national_id_no }}"
OUTPUT "                <span class='error-message'>{{ form.errors.national_id_no }}</span>"
OUTPUT "            </div>"
OUTPUT "            <div>"
OUTPUT "                <label for='{{ form.registration_number.id_for_label }}' class='required'>Student Registration Number:</label>"
OUTPUT "                {{ form.registration_number }}"
OUTPUT "                <span class='error-message'>{{ form.errors.registration_number }}</span>"
OUTPUT "            </div>"
OUTPUT "            <div>"
OUTPUT "                <label for='{{ form.ward_id.id_for_label }}' class='required'>Current Ward of Residence:</label>"
OUTPUT "                {{ form.ward_id }}"
OUTPUT "                <span class='error-message'>{{ form.errors.ward_id }}</span>"
OUTPUT "            </div>"
OUTPUT "        </div>"
OUTPUT "        <div class='form-section'>"
OUTPUT "            <h2>Institution Information Section</h2>"
OUTPUT "            <div>"
OUTPUT "                <label for='{{ form.institution_id.id_for_label }}' class='required'>Institution Name:</label>"
OUTPUT "                {{ form.institution_id }}"
OUTPUT "                <span class='error-message'>{{ form.errors.institution_id }}</span>"
OUTPUT "            </div>"
OUTPUT "            <div>"
OUTPUT "                <label for='{{ form.account_number.id_for_label }}' class='required'>Institution Account Number:</label>"
OUTPUT "                {{ form.account_number }}"
OUTPUT "                <span class='error-message'>{{ form.errors.account_number }}</span>"
OUTPUT "            </div>"
OUTPUT "        </div>"
OUTPUT "        <div class='form-section'>"
OUTPUT "            <h2>Financial Year Information Section</h2>"
OUTPUT "            <div>"
OUTPUT "                <label for='{{ form.financial_year_id.id_for_label }}' class='required'>Current Financial Year:</label>"
OUTPUT "                {{ form.financial_year_id }}"
OUTPUT "                <span class='error-message'>{{ form.errors.financial_year_id }}</span>"
OUTPUT "            </div>"
OUTPUT "        </div>"
OUTPUT "        <div class='form-buttons'>"
OUTPUT "            <button type='submit' class='submit-button blue-button'>Submit</button>"
OUTPUT "            <button type='button' class='red-button' onclick='cancelForm()'>Cancel</button>"
OUTPUT "            <button type='button' class='gray-button' onclick='clearForm()'>Clear</button>"
OUTPUT "        </div>"
OUTPUT "    </form>"
OUTPUT "</div>"
OUTPUT "<script src='{% static 'js/application_form.js' %}'></script>"
OUTPUT "{% endblock %}"
```

7.8.10 Progress Report

```
OUTPUT "{% extends 'base.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<h2>Progress Report</h2>"
OUTPUT "<!-- Display progress report details here -->"
OUTPUT "<form method='post' action='{% url 'progress_report' %}' class='report-form' id='reportForm' data-landing-page-url='{% url 'landing_page' %}'>"
OUTPUT "    {% csrf_token %}"
OUTPUT "    <label for='serial_number'>Serial Number:</label>"
OUTPUT "    <input type='text' name='serial_number' class='serial-input' required placeholder='Enter Serial Number'>"
OUTPUT "        <div class='form-buttons'>"
OUTPUT "            <button type='submit' class='submit-button blue-button'>Generate Report</button>"
OUTPUT "            <button type='button' class='red-button' onclick='cancelForm()'>Cancel</button>"
OUTPUT "            <button type='button' class='gray-button' onclick='clearForm()'>Clear</button>"
OUTPUT "        </div>"
OUTPUT "</form>"
OUTPUT "<script src='{% static 'js/progress_report.js' %}'></script>"
OUTPUT "{% endblock %}"
```

7.8.11 Success Page

```
OUTPUT "{% extends 'base.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<div class='success-container'>"
OUTPUT "    <h2 class='success-title'>Application Submitted Successfully!</h2>"
OUTPUT "    <p class='serial-number'>Your unique serial number is: <b>{{ serial_number }}</b></p>"
OUTPUT "    <p class='serial-number'>Make sure you note it down somewhere for future use!</p>"
OUTPUT "    <p class='serial-number'>You'll need it to track the progress of your application.</p>"
OUTPUT "    <a href='{% url 'landing_page' %}' class='text-link'>Return to Home</a>"
OUTPUT "</div>"
OUTPUT "{% endblock %}"
```

7.8.12 Error Page

```
OUTPUT "{% extends 'base.html' %}"

OUTPUT "{% load static %}"

OUTPUT "{% block content %}"
OUTPUT "<div class='error-container'>"
OUTPUT "    <h2 class='error-title'>Report Generation Failed!</h2>"
OUTPUT "    <p class='serial-number'>Bursary application for the provided serial number does not exist.</p>"
OUTPUT "    <a href='{% url 'landing_page' %}' class='text-link'>Return to Home</a>"
OUTPUT "</div>"
OUTPUT "{% endblock %}"
```

8 Deployment

- Click here to visit bursary mashinani