****

CAMPUSPAY Application

****

**Supervised By**

**Dr. Mohammed Hassan Farrag**

**Presented By**

|  |  |
| --- | --- |
| **Mohamed Ahmed Saad** | **Yousef Ali Saber** |
| **Ahmed Ayman Salah** | **Abdelrahman Mohamed Sayed** |
| **Abdullah Salah Mohamed** | **Abdullah Khaled Sayed** |
| **Esraa Arafa Ahmed** | |

Abstract

Our project outlines the development of a specialized mobile application **"CampusPay**" catering to the unique needs of Fayoum University. **CampusPay** aims to streamline financial transactions for students by introducing a digital wallet and payment system. Key objectives include creating a user-friendly mobile application, implementing a secure digital wallet for bill payments and money transfers, and offering a comprehensive range of services within the app.

**Features and Services**

**Digital Wallet:**

Each student will possess a personalized digital wallet.

Secure loading of funds through various payment methods.

Functionality for both bill payments and money transfers.

**Bill Payments:**

Convenient payment of various college-related bills.

Supported payments include tuition fees, library fines, and printing services.

Real-time notifications and reminders for upcoming bills.

**Money Transfers:**

Easy transfer of funds between student digital wallets.

Facilitates seamless transactions for group projects, events, or personal expenses.

**Implementation Timeline**

**Phase 1**: App Development. (3 months)

**Phase 2**: Integration of Digital Wallet and Services. (2 months)

**Phase 3**: Testing, Quality Assurance, and Bug Fixing. (1 month)

****

**CHAPTER** **ONE**

**INTRODUCTION**

# Chapter One

## Introduction

Our project aims to develop a portable application **(CampusPay**) dedicated to the University of Fayoum, which will be an integrated center for all financial services and payments related to the University of Fayoum and Egyptian universities. This application will streamline and improve payment processes for various university services, both securely and conveniently. **CampusPay** offers all the services of students specificly to the University of Fayoum, from the expenses of study and medical examinations to the payment of expenses for the ability’s exams, University City, registration of courses, and more. **CampusPay** combines all these services into one easy-to-access interface. The app provides a specialized interface to manage student payments. The university can easily track student payments through the Dashboard, where you can find out how many students have paid, collect money, and more. This helps to enhance transparency and facilitate financial procedures. The app also includes a chat to facilitate communication between students and the university. Students can ask their queries and communicate with the university easily, enhancing interaction and bringing the university closer to its students. We are committed to providing an improved and distinctive learning experience for the university and students. Through this application, we will make financial matters easier and more transparent, contributing to excellence in higher education. In addition, the app provides detailed information about tuition fees and other expenses related to the university. Students can view payment details.

## System Overview

**CampusPay** aims to provide a convenient and secure platform for all students at Fayoum University to manage and pay for all their bills and expenses related to the college.

The **CampusPay** app will be equipped with a range of Features, including custom digital wallets for students, support for various payment methods to add money to wallets, and the ability to manage invoice payments and money transfers.

Students can pay for all the services related to the University of Fayoum, such as expenses, medical screening, payment of the expenses of the capacity examinations, the application of courses, the expenses of the sports tourism programmer, courses, graduate registration, graduate services, university cities, and other services.

Additionally, **CampusPay** will offer reminders and real-time bill notifications. Students can also effortlessly transfer money across their digital wallets, making it possible to transfer money for group projects, activities, or personal costs in a smooth manner.

**CampusPay** will be created over time in stages, starting with app creation and ending with testing and bug fixes.

The project's anticipated budget covers development costs, infrastructure installation, and ongoing maintenance. The Digital Wallet and Payments software will, overall, simplify financial transactions for students and give them a convenient and all-inclusive platform for budgeting.

We have concentrated on adding a unique feature, **a machine learning model** that provides secure digital transactions, to our project. For financial transactions, we have used a method known as fraud detection to do this. We can protect users from fraud thanks to this cutting-edge technology and maintain the integrity of transactions. We can analyze patterns and spot any potentially fraudulent behavior by using machine learning algorithms, allowing us to take the necessary precautions to stop any financial losses.

## MOTIVATION

Customizing **CampusPay** for Fayoum University payment services such as tuition, medical fees, SAT fees, course registration, sports tourism program expenses, environmental guidance program expenses, College-organized tourist trips, university city services, graduate registration, and other services. The motivation behind the development of the digital portfolio and the **CampusPay** application of payments to all students at the University of Fayoum and Egyptian universities stems from the need to provide students with a modern and effective solution to manage their financial transactions. Traditional methods of invoice payments and money transfers can be time-consuming, uncomfortable, and prone to errors. Through the introduction of a mobile application designed specifically for the University of Fayoum and Egyptian universities, we aim to streamline these processes and enhance the overall experience of students.

In addition, the introduction of the digital portfolio can radically transform how students deal with their financial obligations. Through the ability to transfer funds safely using a variety of payment methods, students will not have to carry paper money or rely on cash payments. This change is not only a boost to comfort, it also ensures financial security by reducing the risk of theft and loss.

The **CampusPay** application will also provide a range of services that take advantage of the digital wallet function. Students will be able to easily pay a variety of bills related to the University of Fayoum, such as graduate fees, library fines, and printing services, all with a few clicks on their mobile phones. In real time, notices and alerts will keep students informed of their financial obligations and help them avoid additional fees or delayed penalties.

Users expect secure and reliable digital payment systems, and a fraud detection model will therefore be introduced for this application.

## Problem Definition

The current financial transaction and payment management system and the provision of services to students at the Fayoum University are old and ineffective. Students face many challenges when it comes to paying bills and managing their expenses, such as paying expenses, medical screening, paying for capacity exams, soliciting courses, sports tourism programmed expenses, courses, and more services provided by the University of Fayoum.

**1. Inconvenience:** The traditional methods of bill payments and money transfers require students to visit different offices or locations on campus, which can be time-consuming and inconvenient. This process often involves waiting in long queues and filling out multiple forms.

**2. Lack of Real-time Information:** Students often struggle to stay updated on their financial obligations, leading to missed payments, late fees, and penalties. The lack of real-time bill notifications and reminders makes it difficult for students to manage their expenses effectively.

**3. Inefficient Money Transfers:** Collaborative projects, group events, and personal expenses often require students to transfer money between each other. The current system does not provide a seamless and secure method for students to transfer funds, resulting in delays and complications.

**4.** **The student’s inability to pay university fees:**

**5. Lack of Security:** Carrying cash or using physical payment methods puts students at risk of theft or loss. Additionally, the manual handling of cash and paper-based transactions increases the chances of errors and fraudulent activities.

**6.**

7.

8.

## Objectives

Provide an appropriate and safe payment tool for students at Fayoum University, allowing them to pay for all services at Fayoum University and Egyptian universities.

* **Fast services:** In this app, we offer all services for Fayoum University students in an easy way. They can use their wallet to pay for any expenses related to any college in Fayoum University. Each student has a wallet that can be used for payment of expenses.
* **Convenience**: The app will offer a user-friendly interface that allows students to easily pay bills and manage their expenses from their mobile devices. It will eliminate the need for physical visits to different offices or locations on campus, saving students time and effort.
* **Real-time Information**: The app will provide students with real-time notifications and reminders for their financial obligations. They will receive timely updates on bill due dates, payment confirmations, and any changes or updates related to their financial transactions.
* Easy **Money Transfers :**
* **Denators** :
* **Security**: The app will provide a secure platform for financial transactions, reducing the risk of theft or loss associated with carrying cash or using physical payment methods. It will incorporate robust security measures to protect students' financial information and prevent fraudulent activities.
* **Chatting**: The app also includes a chat to facilitate communication between students and the university.
* **Dashboard:** offer to the university clear visions of how many students paid and did not pay and the total amount they received, both from expenses, e-books, courses, and all other services.

**1.5 CampusPay: Users**

|  |  |  |  |
| --- | --- | --- | --- |
| **Students** | **Administrators** | **Moderators** | **Donors** |
| Each student of Fayoum University can pay for all university-related services, including:   * Tuition * Medical fees * SAT fees * Courses registration * Sports tourism * Program expenses * College-organized tourist trips * Graduate registration, and other services | Administrators are responsible for  - Managing, organizing , maintaining the system.  - They have more power than other users, which allows them to perform these tasks effectively. | - Moderators can view data but cannot change it.  - They can see the overall progress of student payments and any outstanding balances.  - They can also see a statistical analysis of payment transactions, which can be used to derive insights that can be used in other relevant processes.  - Moderators may contact the administration to get more information about a specific student. | - Donors can see the social status file of students who are unable to pay university fees.  - They can bear the tuition fees for those students.  - They can also have a conversation with the moderators about the students. |

Table 1.1: CampusPay users

## **CampusPay Features**

### Admin

* **Management of Accounts**: The administrator creates and manages the accounts of users and merchants in the digital payment system. It handles registration, identity verification, account permissions, and access management.
* **Process and Issue Monitoring**: The Administrator monitors digital payment processes, analyzes data, and monitors performance it works to identify problems and disturbances in the system and take the necessary measures to resolve them and ensure continuity of service.
* **Providing technical support**: The administrative officer provides technical support to users and merchants regarding digital payment operations. He responds to inquiries, solves problems, and guides users in using the system correctly.
* **Improving and developing the system:** Listens to user feedback and suggests the necessary improvements to enhance the user experience and facilitate payment processes.
* **Managing reports**: Admins generate reports on all activities and transactions that occur within the app and send them to the manager. These reports provide valuable insights into the app's performance, user behavior, and overall Transaction.
* **Chat moderator :**

### Student

* **Pay all services:** There are various services available for students to pay for, such as study expenses, e-books, university housing expenses, medical examination expenses, tuition fees for course registration, and more. These services are designed to assist students in managing their financial obligations. Students have the option to make payments using the CampusPay wallet, providing them with convenient and secure payment methods.
* **Deposit Funds:** The student can deposit funds into his digital account using the payment methods supported by the system. These methods can include wire transfers, the use of credit cards, or wallets.
* **Review Balance and Records:** By reviewing his balance and transaction records in his digital account, the following transactions are reached: the amounts received or sent are verified, and the remaining balances are counted.
* **Chat with Moderator:** The student can contact the student support team in case of problems or inquiries.

### Moderator

* **Review Dashboard**: The moderator can access the Dashboard, which displays all existing statistics and provides useful information.
* **Chat with admin:** where, if there are any problems in the Dashboard, the admin can modify it.
* **Chat with Donor :**
* **Chat with students :**

### Donor

* **View student's social status file**: A donor can see the social status file of students who are unable to pay university fees.
* **covering the tuition fees for students**: A donor can cover the tuition fees for a student who is unable to pay university fees.
* **Chat with Moderator:** A donor can have a conversation with the moderators about those students.

****

**CHAPTER TWO**

**Literature Review**

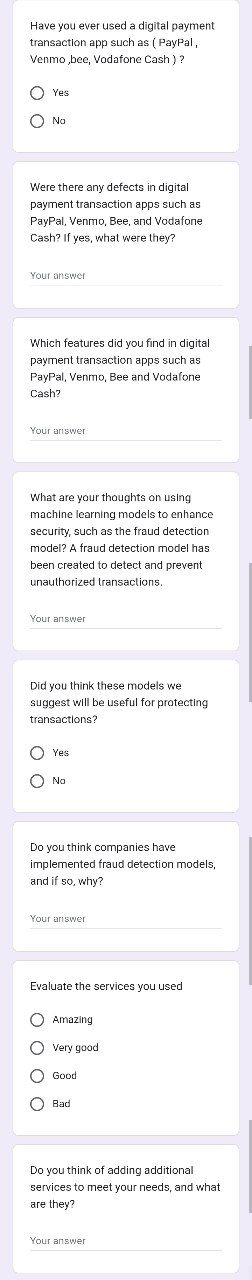
# Chapter two

## INTRODUCTION:

The successful implementation of the **CampusPay** app hinges on thorough project planning, encompassing objective and requirement definition, task allocation, resource identification, and cost estimation. This comprehensive approach will facilitate the development of an integrated digital payment system for Fayoum University students. Moreover, a machine learning model will be employed to safeguard transactions, with a focus on fraud detection.

## Project Planning

* **Gather requirements for digital payment transactions**
* **System requirements**: Requirements for the digital payment system must be defined and documented. This includes system functions such as creating user accounts, transactions, and providing financial estimates.
* **Interface requirements**: User interface requirements must be precisely defined. This includes identifying attractive design, ease of use, and clarity of operations and direction.
* **Services requirements**: Here we need to offer all digital payment services for our students at Fayoum University.
* **Fraud detection Model requirements**: Requirements for a fraud detection model for digital financial transactions must be defined. This includes using statistical analysis and machine learning techniques to detect patterns of fraud and suspicious behavior.

 Figure 2.2: questionnaire

## Feasibility study

### Conducting a feasibility study is a crucial step before embarking on any technology project. This assessment determines whether the project is viable, economically sustainable, and aligned with achieving the desired objectives. In the case of the CampusPay app, which aims to provide comprehensive electronic payment solutions for Fayoum University students' tuition fees, a business case analysis helps evaluate the potential benefits, costs, and risks associated with its implementation.

### Executive Summary

**CampusPay** provides ample opportunities to facilitate payment processes for our students and improve the efficiency of financial operations. With the increasing use of digital technology worldwide, detecting fraud in financial transactions has become an urgent necessity.

### Description of products and services

**CampusPay** enables Fayoum University students to use digital payment services for money transfers and bill payments through Wallet. The app also includes a fraud detection model that ensures secure and reliable financial transactions.

### 2.3.4 Organization and Staffing

Include an integrated team each of whom has an effective role. Positions are distributed according to each person’s ability to perform the assigned tasks.

|  |  |
| --- | --- |
| **Team Leader** | Who manages, distributes tasks, handled team members, solve internal problems, and leads the team well. |
| **Data Science & Machine Learning** | Data scientists collect, clean, and analyze required data, develop explanatory models, and extract knowledge from data through techniques such as clustering, regression, and data classification. |
| **Back-end Developer** | Who is building and maintaining the server-side of the application. They are responsible for designing and implementing the application's database structure, handling data storage and retrieval, and managing the app's server infrastructure. |
| **Flutter Developer** | Who is responsible for the user interface design in mobile applications. |
| **UIUX Designer** | UI stands for User Interface, and UX stands for User Experience. UI is the visual and interactive elements of a product that users interact with, such as the buttons, menus, and text. UX is the overall experience that a user has when using a product, including how easy it is to use, how enjoyable it is to use, and whether it meets the user's needs |
| **Documenter** | Who documents all the steps that happen in the project |

Table 2.1: Organization and Staffing.

### Service Marketplace

Fayoum University students can easily access digital payment services, transfer money, and pay bills through a mobile application called Wallet. The app comes with a fraud detection model that ensures secure and reliable financial transactions.



Figure 2.4: Project Team

## Risk IDENTIFICATION

Risk identification is a critical process that uncovers and evaluates potential threats to operations.

## Schedule RISK

Poor planning can lead to schedule risk, which is the risk that project tasks and activities will take longer than expected to complete. This risk is closely tied to cost risk, as any delays in the schedule can result in increased costs, slower project benefits, and disrupted timelines, ultimately leading to the loss of any competitive advantage that was initially gained.

## Performance Risk

**CampusPay**’s performance and reliability during transactions can be affected by various issues. These issues include:

* slow response times
* frequent downtime or system failures
* scalability issues
* poor network connectivity
* inefficient data processing

To avoid these issues, the app should:

* optimize performance
* have robust server infrastructure
* conduct load testing
* handle network interruptions
* optimize data processing

## Operational Risk

The app may face some challenges that can affect how it works, how reliable it is, and how secure it is. These challenges are:

* system outages
* transaction errors
* technical glitches
* fraudulent activities
* inadequate customer support

To prevent these challenges, the app needs to:

* implement redundancy measures
* use robust validation mechanisms
* do regular testing
* apply strong security measures
* provide efficient customer support
* monitor transaction data

## Strategic Risk

The app needs to have a good strategy and method to find and stop fraudulent activities. Some of the main risks are:

* not having good fraud detection capabilities
* having false positives and negatives
* facing new fraud techniques
* having integration challenges

To reduce these risks, the app should:

* invest in advanced technologies
* optimize algorithms
* stay updated on fraud trends
* ensure seamless integration

It is also important to monitor and analyze how well the fraud detection works.

## Legal Risk

The app has to follow the laws, rules, and contracts that apply to it. Some of the main risks are:

* protecting the privacy and data of the users.
* following the regulations that apply to the app.
* meeting the contractual obligations with the partners.
* not infringing on the intellectual property of others.

****

**CHAPTER TREE**

**Project Management**

# Chapter three

## overview

This chapter delves into the historical context of digital payment transaction technology, examines the role of systems and platforms in streamlining payment processes, and evaluates both the strengths and limitations of these solutions

## Historical background

### For digital payment transaction

* The first major milestone in the history of digital payment transactions was the creation of the Automated Clearing House (ACH) network in the United States in the early 1970s. The ACH network allowed for electronic funds transfers between banks, enabling businesses and individuals to make payments directly from their bank accounts.
* The 1990s saw the emergence of online payment systems, such as PayPal, which revolutionized the way people conducted transactions over the internet. These systems allowed individuals to transfer money digitally, making online shopping and e-commerce more accessible.
* The early 2000s witnessed further advancements in digital payment technology with the introduction of mobile payment systems. Companies like Apple, Google, and Samsung developed mobile payment platforms that allowed users to make payments using their smartphones, further enhancing convenience and accessibility.

### For fraud detection model

* In the early stages of digital payment systems, fraud detection primarily relied on manual processes and human intervention. Banks and financial institutions would manually review transactions for any suspicious activity or discrepancies. However, as the volume of digital transactions increased, this approach became inefficient and time-consuming.
* The introduction of automated fraud detection systems in the late 20th century marked a significant advancement in combating payment fraud. These systems utilized algorithms and data analysis techniques to identify patterns and anomalies that could indicate fraudulent activity. Machine learning and artificial intelligence technologies were employed to continuously learn and adapt to new fraud patterns, improving the accuracy of fraud detection.
* As digital payment methods evolved, so did the methods used by fraudsters. The rise of online shopping and e-commerce led to new types of fraud, such as account takeovers, identity theft, and card-not-present fraud. To address these challenges, fraud detection systems incorporated additional security measures, such as two-factor authentication, IP geolocation, and device fingerprinting.
* In recent years, advancements in technology, such as big data analytics and real-time monitoring, have further enhanced fraud detection capabilities. These technologies enable the analysis of vast amounts of data in real-time, allowing for the detection of fraudulent transactions as they occur.

## Brief History of Using Technology for digital payment transaction

* The use of technology in digital payment transactions has a relatively short but impactful history. It began with the development of electronic payment systems in the late 20th century and has since evolved rapidly with technological advancements.
* In the 1970s, the introduction of credit cards laid the foundation for digital payment transactions. These cards allowed consumers to make purchases without the need for cash, and transactions were electronically processed through payment networks.
* The next significant development came in the 1990s with the emergence of online shopping and e-commerce. This period saw the rise of online payment gateways and the introduction of secure protocols for online transactions. Companies like PayPal, founded in 1998, played a crucial role in enabling individuals and businesses to make and receive payments electronically.
* The early 2000s witnessed the widespread adoption of mobile phones and the advent of mobile payment technologies. Initially, mobile payments were limited to basic features like SMS-based payments, but with the introduction of smartphones, more advanced mobile payment solutions emerged. Companies like Apple with Apple Pay and Google with Google Pay introduced digital wallet systems that allowed users to make payments using their smartphones.

## Technological Digital Payment TRANSACTION SYSTEMS

* A technological digital payment system refers to the use of electronic devices and platforms to facilitate financial transactions. This system eliminates the need for physical cash and allows individuals and businesses to make payments or receive funds electronically such as Mobile Payment Systems These systems use mobile devices such as smartphones or tablets to process payments. Examples include mobile wallets such as Apple Pay, Google Pay, and Samsung Pay by using the fraud detection model to detect and prevent unauthorized transactions is crucial to mitigate financial losses.
* Based on much research, there are many proposals to improve the digital payment transaction. according to much research, the proposed systems can be classified.

**CampusPay System**

Mobile Application

Online payment

Payment cards

* Peer to Peer Fund Transfer
* BIII Payment
* Electronic Fund Transfer
* Bill Payments
* Credit Cards
* Debit Cards
* Prepaid Cards

## Related System

1. **PayPal System** 
   * stands as a leading online payment system, functioning as a versatile payment gateway for users worldwide. Offering various account types, it allows individuals and businesses to link their accounts to credit cards, debit cards, and bank accounts. With support for international transactions in multiple currencies, PayPal is a global platform. It provides buyer and seller protection, secure Express Checkout, and tailored business solutions. Notably developer-friendly, PayPal furnishes robust APIs and SDKs, ensuring seamless integration into websites and applications.
2. **Vodafone Cash**
   * a mobile financial service by Vodafone, offers a convenient and secure platform for various financial transactions. Acting as a virtual wallet on users' mobile devices, it enables seamless money transfers, bill payments, and mobile recharges. With features like airtime top-up and merchant payments, Vodafone Cash promotes a cashless economy and simplifies day-to-day transactions. The system incorporates security measures such as PIN codes and SMS-based notifications, ensuring the safety of users' financial activities.
3. **myFawry**
   * is a digital payment platform that streamlines financial transactions for users in Egypt. Serving as a comprehensive financial solution, myFawry allows individuals to conveniently pay bills, purchase goods and services, and even recharge mobile credit through a user-friendly mobile app. The platform's versatility extends to supporting various payment methods, enhancing accessibility for a wide range of users.
4. **Rules-based Systems**

* Rules-based systems use predefined rules and thresholds to flag transactions that meet certain criteria for potential fraud. These rules can be based on factors such as transaction amount, location, or frequency.

1. **Biometric Authentication**

* Biometric authentication technologies, such as fingerprint or facial recognition, can be used to verify the identity of individuals during financial transactions. This helps prevent unauthorized access and fraudulent transactions.

1. **Fraud Analytics Platforms**

* Fraud analytics platforms integrate various technologies and techniques to provide comprehensive fraud detection capabilities. These platforms often combine machine learning, behavioral analytics, and rules-based systems to analyze transaction data and identify potential fraud.

1. **Transaction Monitoring Systems**

* Transaction monitoring systems analyze individual transactions in real-time and compare them against known patterns of fraudulent activity. These systems can flag suspicious transactions for further investigation.

1. **Data Visualization**

* Data visualization techniques are used to present complex financial transaction data in a visual format, making it easier to identify patterns or anomalies that may indicate fraudulent activity.

## **CampusPay System**

### Advantages

* **Convenience and Ease of Use**

CampusPay utilizes digital payment methods, such as mobile wallets and online banking, to provide users with a convenient and accessible payment experience. Transactions can be made anytime, anywhere, without the need for physical cash or cards.

* **Centralized Payment Platform**

**CampusPay** consolidates all student payment needs into a single platform, eliminating the hassle of managing multiple accounts and payment methods. This centralized approach streamlines the payment process for tuition fees, e-book expenses, university housing expenses, and more.

* **Enhanced Security**

**CampusPay** prioritizes the security of sensitive financial information by incorporating robust measures such as encryption and multi-factor authentication. Additionally, fraud detection models continuously monitor transactions for suspicious activities to safeguard against unauthorized access and fraudulent transactions.

**CampusPay** offers students a secure, convenient, and customized payment solution, enhancing their overall university experience.

### Weaknesses

* **False Positives in Fraud Detection**

**CampusPay**'s fraud detection models, while designed to identify and prevent fraudulent transactions, may occasionally generate false positives, incorrectly flagging legitimate transactions as suspicious. This can lead to inconveniences for users, such as delayed transactions or account restrictions, and may even cause missed opportunities.

* **Data Privacy and Security Concerns**

The use of digital payment systems necessitates the collection and processing of personal and financial data. While CampusPay implements robust security measures to protect this data, concerns about privacy and potential breaches may persist among users.

**The technologies used to develop CampusPay**

|  |  |
| --- | --- |
| * **Flutter** | A cross-platform framework for building user interfaces for mobile, web, and desktop applications. It uses a declarative approach to create widgets that are responsive, expressive, and fast. |
| * **UI UX** | **UI** stands for User Interface, and **UX** stands for User Experience. UI is the visual and interactive elements of a product that users interact with, such as the buttons, menus, and text. UX is the overall experience that a user has when using a product, including how easy it is to use, how enjoyable it is to use, and whether it meets the user's needs. |
| * **Machine learning** | A branch of artificial intelligence that enables computers to learn from data and make predictions or decisions. |
| * **Python** | A popular language for machine learning because it has many libraries and tools that support data analysis, visualization, and model development. |
| **Data Analysis** | A process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, and is used in different business, science, and social science domains helping businesses operate more effectively. |
| **.NET framework** | A software platform that provides a set of tools and libraries for building and running applications on Windows, Linux, and macOS. It supports multiple programming languages, such as C#, F#, and Visual Basic. |
| * **CSharp** | A general-purpose, object-oriented programming language developed by Microsoft. It is pronounced "C sharp" and is one of the most popular programming languages in the world. C# is used to develop a wide variety of applications, including desktop applications, web applications, mobile applications, and games. |
| **Microsoft SQL Server** | A relational database management system (RDBMS) that supports a wide variety of transaction processing, business intelligence, and analytics applications in corporate IT environments. It is one of the most popular RDBMSs in the world, and it is used by millions of businesses and organizations of all sizes. |

The technologies used to develop CampusPay

**CampusPay Architecture**

**FLUTTER**

**UI, UX**

**Presentation**

**MACHINE LEARNING**

**DATA ANALYSIS**

**(WITH PYTHON)**

**Business**

**MSSQL**

**.NET FRAMEWORK**

**(WITH C#)**

**Data Access**

**Data** **Storage**

|  |  |
| --- | --- |
| **Other Digital payment apps** | **CampusPay** |
| Digital payment apps such as MyFawry, Vodafone Cash, PayPal, and others are **general-purpose payment apps** that offer a wide range of payment services, including money transfers, cash withdrawals, bill payments for utilities such as water, gas, electricity, and other household bills.  They also offer **a tuition** payment service, but it is a general service that is **not customized** for other university services. | CampusPay is designed **specifically** for Fayoum University students, with the potential to be expanded to include all universities in Egypt in the future.  CampusPay is limited to payments that students need to make, such as tuition fees or student services. This means that it is not a general digital payment app like other digital payment apps, but rather a **customized** app for Fayoum University students. |

****

**Chapter 4**

**System Analysis**

## System overview

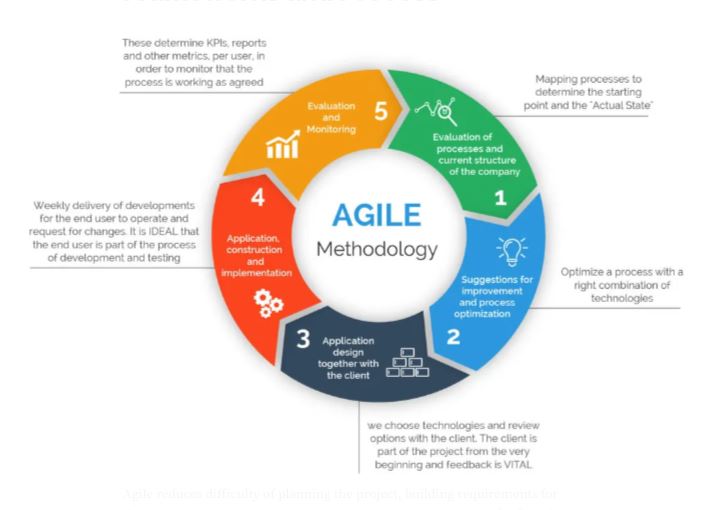
This chapter reviews the Development methodology, the Agile Software Development Life Cycle also reviews the functional and non-functional requirements. It also discusses System analysis through the life cycle and its requirements showing the USE-CASE Diagram which is a list of actions or events steps typically defining the interactions between a role known in the Unified Modeling Language as an actor and a system to achieve a goal, Activity diagram which is a graphical representation of workflows of stepwise activities and actions and process modeling.

## **Development methodology**

Standardized process followed in an organization to conduct all the necessary steps for analyzing, designing, implementing, and maintaining information systems. It provides a structured approach to ensure efficient and effective development of software or information systems.

**Agile Methodology**

The Agile methodology emphasizes flexibility, collaboration, and iterative development. It involves breaking the development process into smaller iterations called sprints, where features are developed, tested, and delivered incrementally.

 Figure 4.2: Agile methodology

## The Agile Software Development Life Cycle

* It is the combination of both iterative and incremental process models. It focuses on process adaptability and student satisfaction by rapid delivery of working software product. Agile SDLC breaks down the product into small incremental builds. These builds are provided into iterations.
* The Agile software development life cycle consists of six phases: concept, inception, iteration, release, maintenance, and retirement.

**Phase1 Concept**

This is the first phase in the agile SDLC where we define the project scope. If there are numerous problems, they will prioritize the most important ones. It also defines the project objectives, key requirements, and estimates the cost and time of the project. This detailed analysis will help them to decide whether a project is feasible before commencing work

**Phase2: Inception**

Once the concept is outlined, it is time to start the design process and build the project architecture. This stage involves further input to fully flesh out the requirements on a diagram

**Phase3: Iteration**

It also referred to as construction. It tends to be the longest phase that is carried out here. The goal is to build the first draft functionality of the project by the end of the first iteration. This stage is a cornerstone of agile software development, enabling developers to create working software quickly and make improvements to satisfy the client.

**Phase4: Release**

The project is almost ready for release in this stage. But first, we need to perform some tests to ensure the software is fully functional. User training will also take place during this phase, which will require more documentation. When all of this is complete, the project’s final iteration can then be released into production.

**Phase5: Maintenance**

The software will now be fully deployed and made available to students. So, during this phase, we should provide ongoing support to keep the system running smoothly and resolve any new bugs. Over time, new iterations can take place to refresh the existing project with upgrades and additional features.

**Phase6: Retirement**

There are two reasons why a product will enter this phase: either it is being replaced with new software, or the system itself has become has become obsolete or incompatible with the organization over time. The team will notify users if the software is retired and they will be migrated to the new system.

## Process Modeling

### **Context Diagram**

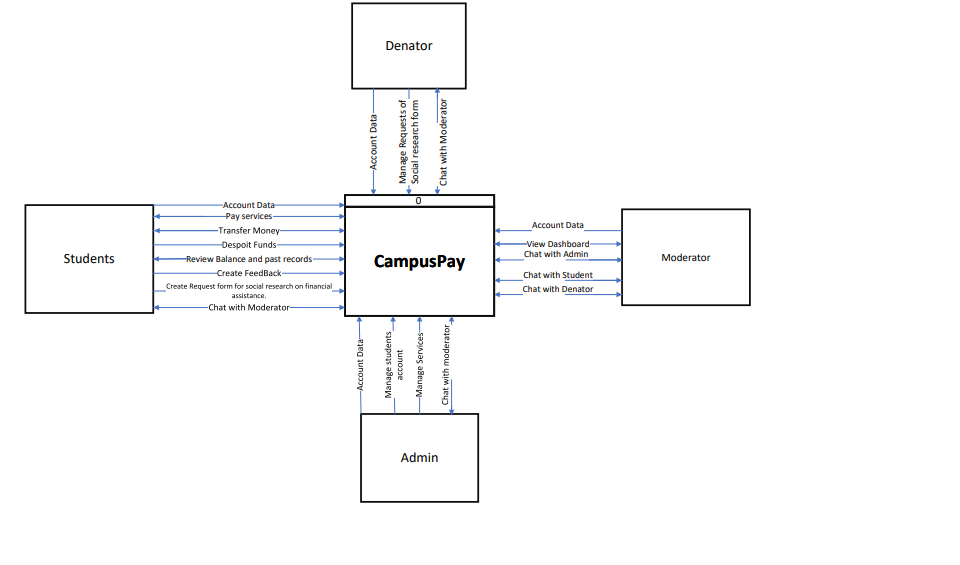
Context Diagram is a diagram that defines the boundary between the system, and its environment, showing the entities that interact with it. This diagram is a high-level view of a system. As a result, Context Diagram can figure out how best to design a new system and its requirements or how to improve an existing system.

Figure 4.3: context diagram

## System Requirements

### Functional Requirements

Functional requirements are features that allow the system to function as it was intended. Put another way, if the functional requirements are not met, the system will not work. Functional requirements are product features and focus on user requirements.

* **Denator**

For Denator these functions include

* + Login Functionality
  + Manage Requests of Social research Form
  + Chat with Moderator
* **Admin**

For Admin these functions include

* + Login Functionality
  + Mange Services
  + Mange Accounts
  + Chat with Moderator
  + Logout Functionality
* **Moderator**

For Moderator these functions include

* + Login Functionality
  + Interact with Dashboard
  + Chat with Admin
  + Chat with Student
  + Chat with Denator
  + Logout Functionality
* **Student**

For student these functions include

* + Sign up/Sign in Functionality
  + Transfer Money
  + View & Pay services
  + Chat with Moderator
  + Create Feedback
  + Logout Functionality
  + Despoit Funds
  + Review balance and Last Transactions
  + Create a request form for Social research on financial assistance

Though Requirements are usually written in text, especially for Agile-driven projects, however, they may also be visuals. Here are the most common formats and documents:

* Use Case Diagram.
* Use Case Scenario.
* Activity Diagram**.**

### Use Case Diagram

* Use cases represent system functionality from the user’s perspective
* Use Case diagrams describe who will use the system and in what ways the user expects to interact with the system.
* Use Case diagrams represent the interactions between use cases and actors.
* Use Case diagram represents the interactions between system, external systems, and users.

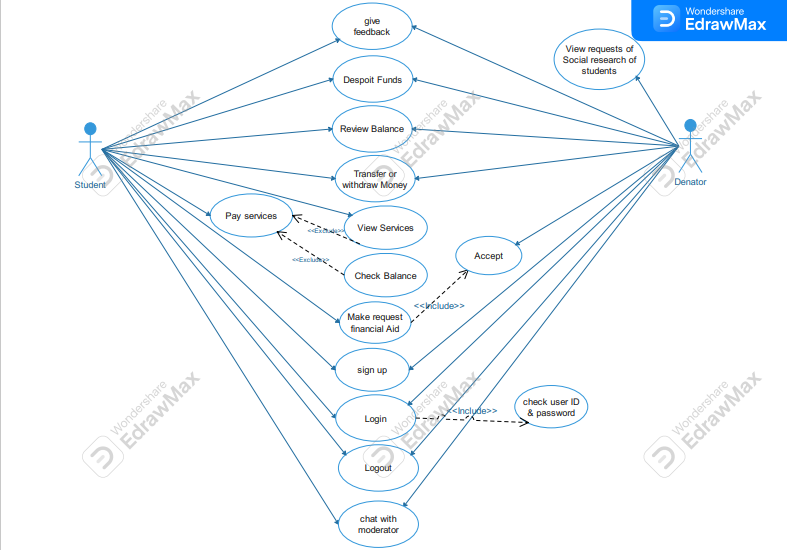


Figure 4.4: Student and Denator Use Case Diagram

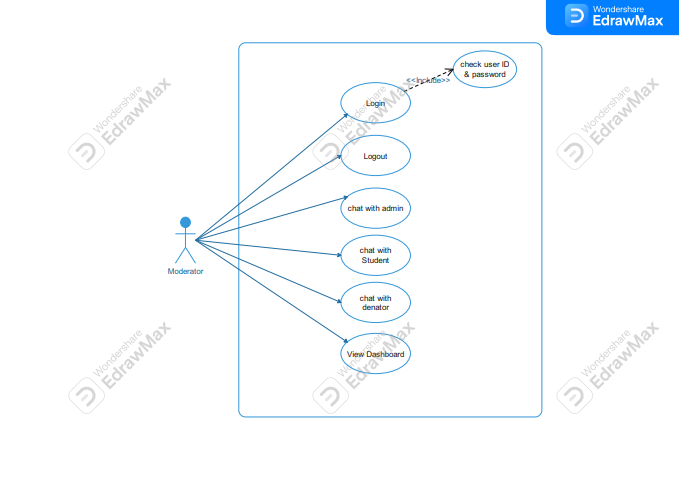


Figure 4.5:Moderator use case Diagram

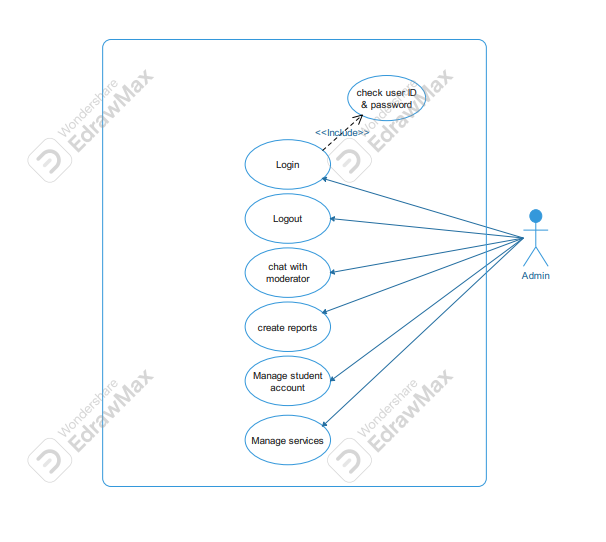


Figure 4.6: Admin Use Case Diagram

### Use Case Scenario :

* A use case Scenario represents the sequence of events along with other information that relates to this use case. A typical use case specification template includes the following information:
  + Description
  + Pre and Post interaction condition
  + Basic interaction path
  + Alternative path.

|  |  |  |
| --- | --- | --- |
| Use case Name: | Manage Accounts | |
| Actor(s): | Admin | |
| Description: | This use case allows admin to delete , edit and update accounts | |
| Typical Course of Events: | Actor Action | CampusPay app Response |
|  | **Step** 1 : login into the CampusPay app.  **Step3 : admin can navigate to account section**  **Step 4:** admin can delete , edit , update accounts | **Step 2:**  The CampusPay app view the main interface of the admin  **Step4 : the campusPay app will view all accounts**  Step 5 : CampusPay app execute admins action |
| Alternate Courses: | - If the admins encounters an issue while modifying an account, an error message is displayed.  If the system cannot disable or remove an account due to dependencies, the admin is notified. | |
| Precondition**:** | -The admin must have valid login credentials. | |
| Post condition: | -User account information is updated as per the admin's actions.  -Changes made by the admins are reflected in the CampusPay app. | |
| Assumption: | - The admin can manage actor accounts using the CampusPay app.  -The system has appropriate security measures to protect user account information.  -The admin understands the implications of the actions taken on user accounts. | |

Figure 4.7: Manage Accounts Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | Manage services | |
| Actor(s): | Admin | |
| Description: | This use case allows the admin to add, delete, or edit a specific service | |
| Typical Course of Events: | Actor Action | CampusPay app Response |
|  | **Step** 1 : login into the CampusPay app.  **Step3 : admin can navigate to service section**  **Step 5:** admin can add , remove , edit a specific service | **Step 2:**  The CampusPay app offers special privileges to admin  **Step4 : the campusPay app will view all services**  **Step 6** : CampusPay app execute admins action |
| Alternate Courses: | - If the admins encounters an issue while adding or modifying an services, an error message is displayed.  - If the system cannot disable or remove a service due to dependencies, the admin is notified. | |
| Precondition**:** | -The admin must login and have all privileges to manage services. | |
| Post condition: | -a specific service is updated or added or deleted as per the admin's actions.  - Changes made by the admins are reflected in the CampusPay app. | |
| Assumption: | - The admin can manage all services using the CampusPay app.  -The admin understands the implications of the actions taken on these services  -The system is connected to a reliable user database. | |

Figure 4.7: Manage Services Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | **View Dashboard** | |
| Actor(s): | Moderator | |
| Description: | This use case allows Moderators to use the CampusPay app to view the dashboard or a specific report to see what is happening in this app. This will allow them to make an informed decision that will help them to improve their services or something else. | |
| Typical Course of Events: | Actor Action | CampusPay app Response |
|  | **Step 1:** Moderator will sign in CampusPay app  **Step 3:**  The moderator reviews on the dashboard to extract some insights | **Step 2: the CampusPay app will view the dashboard which is the main interface**  **Step 4:** The CampusPay app provide a lot of dynamic visuals that connected to DataBase |
| Alternate Courses: | - Dashboard Not viewed : this can happen based on some cases like maybe admin didn’t make any reports , or there is an error in CampusPay app so it must call the Admin | |
| Precondition**:** | - The Moderator is logged into the CampusPay app.  - The moderator can see the Dashboard and he can review it. | |
| Post condition: | - The Moderator make some analysis on Dashboard to extract some insights or make a decision | |
| Assumption: | - The mdoerator is successfully logged into the system.  - The campuspay app efficiently presents the Dashboard. | |

Figure 4.8: View DashboardUse Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | **view requests for social research made by students.** | |
| Actor(s): | Donator | |
| Description: | This use case allows to donators to use the CampusPay app to view social research requests made by students who are seeking financial aid. This will allow them to make an informed decision about whether to accept the request or not | |
| Typical Course of Events: | Actor Action | CampusPay app Response |
|  | **Step 2:** The donator navigate to the “ view requests for social research made by students.” section  **Step 4:**  The donator reviews the list of requests and selects a specific request to view the social research details .  **Step 6 :**The donor can take various actions based on the request, such as accepting, rejecting request | **Step 1:** The CampusPay App presents the student requests  **Step 3:**  The Campus Pay App provides access to view students' requests for social research that students send it .  **Step 5:** The CampusPay App displays the detailed information of the selected request, including the requestor's information and the social research created by student .  **Step 7:** The CampusPay app provides options and actions to the donator based on their decision |
| Alternate Courses: | - error when viewing request : donator can view another requests and if he face problems he can chat with moderator to help him | |
| Precondition**:** | - The donor is logged into the CampusPay app.  - The "View students Request" section is accessible to the donor.  - There are pending requests for assistance or donations. | |
| Post condition: | - The donor can view detailed information about specific requests.  - The donor can take actions based on the requests, such as accepting or rejecting them. | |
| Assumption: | - The donator is successfully logged into the system.  - The system efficiently presents a list of pending requests.  - Request details are accurate and reflect the requestor's needs.  - Donators can take actions on the requests through the dashboard. | |

Figure 4.9: view requests for social research made by students Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | Login | |
| Actor(s): | Student, Admin, Moderator, Denator | |
| Description: | This use case allows a user to log in to a system or application using their credentials. | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 1:** accesses the system or application.  **Step 3:** enters their username and password. | **Step2:** The system presents a login interface.  **Step 4:** The system validates the entered credentials  **Step 6:** The system grants access and displays the user's dashboard or the main interface. |
| Alternate Courses: | The user enters incorrect username and/or password, The system detects the incorrect credentials. | |
| Precondition**:** | The user has registered with the system or application. | |
| Post condition: | The user gains access to the system or application and is directed to their dashboard or the main interface upon successful login. | |
| Assumption: | The user's username and password are stored securely in the system. | |

Figure 4.10: Login Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | Review Balance | |
| Actor(s): | Student | |
| Description: | This use case allows Actors to review their account balance or financial status through the system. | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 1:** navigates to the "Review Balance section.  **Step 3:** views their account balance or financial information. | **Step2:** The system provides access to the balance review interface.  **Step 4:** The system retrieves and displays the current balance, transaction history, or relevant financial details. |
| Alternate Courses: | Account Activity: The student view their account balance but wants to see recent account activity, The system allows the student to access the account activity log. | |
| Precondition**:** | - The Actor is logged into the system.  - The "Review Balance" section is accessible to the Actors. | |
| Post condition: | - The Actor review their account balance or financial status.  - The Actor may choose to perform additional actions like reviewing account activity or inquiring about specific transactions. | |
| Assumption: | - The Actors is successfully logged into the system.  - The system can efficiently retrieve and display financial information.  - The financial information is up-to-date and accurate. | |

Figure 4.11 : Review Balance Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | View Service | |
| Actor(s): | Student | |
| Description: | This use case allows a student to check the availability and details of a particular service. | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 2:** navigates to Service section  **Step 4:** search about a particular service  **Step 6:** The Actor reviews the service details. | **Step1:** The system presents the student with the main interface.  **Step 3:** The system provides access to a service lookup interface.  **Step 5:** The system searches for the service and presents the relevant details.  **Step 7:** The system displays information about the service, including its description, availability, and any associated options. |
| Alternate Courses: | Service Not Found: The Actor must chat with the moderator to solve the problem he is facing. | |
| Precondition**:** | - The student is logged into the system.  - The "Check Service" section is accessible to the student.  - The student has the name or ID of the service they want to check.  - The service lookup system is operational. | |
| Post condition: | The student receives information about the service they checked. | |
| Assumption: | - The student is successfully logged into the system.  - The system can efficiently search for and retrieve service information.  - Service information is up-to-date and accurate.  - The service can be identified by a name or ID.  - The student is familiar with the "Check Service" process. | |

Figure 4.12 : View Service Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | Pay Service | |
| Actor(s): | Student | |
| Description: | This use case allows a student to make a payment for a specific service within the system. | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 2:** The student navigates to the “Services” section.  **Step 3:** The student selects a specific service they want to pay for.  **Step 5:** The student confirms their intention to pay for the service from wallet. | **Step1:** The system presents the student with the main interface.  **Step 4:** The system displays information about the selected service, including the service name, description, and the payment amount.  **Step 6:** The system Check amount of money in wallet and processes the payment and provides a payment confirmation. |
| Alternate Courses: | Payment Declined: The system attempts to process the payment, but it is declined and The system displays an error message if the balance is not enough, but if the service is not working, the student should chat with the moderator about a particular service. | |
| Precondition**:** | - The student is logged into the system.  - The "Pay Service" section is accessible to the student.  - The student has selected a specific service to pay for.  - The system displays information about the selected service, including the service name, description, and the payment amount. | |
| Post condition: | - The student should pay the service. | |
| Assumption: | - The student is successfully logged into the system.  - Payment method are secure and easy.  - The selected service is available for payment.  - The student is familiar with the payment process within the system. | |

Figure 4.13 : pay Service Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | Make Request for Financial Aids | |
| Actor(s): | Student | |
| Description: | This use case allows a student to make a request for financial aids from Donor. | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 2:** The student navigates to the "Make Request for Financial Aids" section.  **Step 4:** The student completes the request form, providing necessary details, such as the reason for the request, financial information, and required aid amount | **Step1:** The system presents the student with their dashboard or the main interface.  **Step 3:** The system provides access to a request social research form.  **Step 5:** The system validates the form entries and confirms the submission.  **Step 7:** The system acknowledges the request and provides a confirmation message. |
| Alternate Courses: | - Request Modification: The student has submitted a request but needs to modify some details , The system allows the student to access and modify the request before it is reviewed.  -Form not available so student should chat with moderator | |
| Precondition**:** | - The student is logged into the system.  - The "Make Request for Financial Aids" section is accessible to the student.  - The student has the necessary information to complete the request form. | |
| Post condition: | - The student's request for financial aids is recorded in the system.  - The student receives a confirmation of the request submission. | |
| Assumption: | - The student is successfully logged into the system.  - The system can handle and process financial aid requests.  - The request form includes validation and error handling.  - The request information is accurately recorded for review. | |

Figure 4.14 : Make Request for Financial Aids Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | Deposit Funds | |
| Actor(s): | Student | |
| Description: | This use case allows a user to deposit funds into their account through the system. | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 2:** The student navigates to the "Deposit Funds" section.  **Step 4:** The student selects the payment method (e.g., credit card).  **Step 6:**  The student enters the deposit amount and confirms the transaction. | **Step1:** The system presents the student with their dashboard or the main interface.  **Step 3:** The system provides access to a deposit interface.  **Step 5:** The system displays the payment options and prompts the student to choose a method.  **Step 7:** The system validates the deposit amount and processes the transaction  **Step 8:** The system displays a confirmation message and updates the student's account balance. |
| Alternate Courses: | - Payment Rejection: The payment transaction is rejected, The system displays an error message with rejection details and allows the student to retry the payment or choose an alternative payment method. | |
| Precondition**:** | - The student is logged into the system.  - The "Deposit Funds" section is accessible to the student.  - The student has chosen a valid payment method.  - The student has a sufficient balance or a valid source of funds for deposit. | |
| Post condition: | - The student's account balance is updated with the deposited funds.  - The payment transaction is recorded in the system. | |
| Assumption: | - The student is successfully logged into the system.  - The system can handle payment methods that users have deposited with it.  - The student has access to a valid payment method.  - The system can provide error messages in case of payment rejection.  - The student is familiar with the deposit funds process. | |

Figure 4.15 : Deposit Funds Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | Transfer money | |
| Actor(s): | Student, Donor | |
| Description: | This use case allows a user to transfer money to  any athers users | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 2:** The users initiates a money transfer to anther  **Step 4:** The student selects the payment method (e.g., credit card, bank transfer).  **Step 6:**  The student enters the deposit amount and confirms the transaction. | **Step1:** The system presents the user with their dashboard or main interface.  **Step 3:** The system provides access to a deposit interface.  **Step 5:** The system displays the payment options and prompts the student to choose a method.  **Step 7:** The system validates the deposit amount and processes the transaction  **Step 8:** The system displays a confirmation message and updates the student's account balance. |
| Alternate Courses: | - Payment Rejection: The payment transaction is rejected, The system displays an error message with rejection details and allows the student to retry the payment or choose an alternative payment method.  - Change Payment Method: The student selects the payment method but decides to change it, The system allows the student to switch to a different payment method before confirming the transaction. | |
| Precondition**:** | - The student is logged into the system.  - The "Deposit Funds" section is accessible to the student.  - The student has chosen a valid payment method.  - The student has a sufficient balance or a valid source of funds for deposit. | |
| Post condition: | - The student's account balance is updated with the deposited funds.  - The payment transaction is recorded in the system. | |

Figure 4.16: Transfer money

|  |  |  |
| --- | --- | --- |
| Use case Name: | Give Feedback | |
| Actor(s): | Student, Denator | |
| Description: | This use case allows students and denators to provide feedback on the services or experiences they have had. | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 2:** The student selects the type of feedback (e.g., service feedback, donation feedback, program feedback, general feedback).  **Step 4:** The student fills out the feedback form, including comments, ratings, and any relevant details. | **Step1:** The System presents the student with the feedback submission interface.  **Step 3:** The System provides a feedback form tailored to the selected type.  **Step 5:** The System validates and stores the feedback.  **Step 7:** The System sends this feedback for admin or moderator Based on type of feedback |
| Alternate Courses: | - Feedback Modification: The student or denator wants to edit previously submitted feedback, The System allows the user to locate and edit their feedback entries.  - Feedback Deletion: The student or denator wants to delete previously submitted feedback, The System allows the user to locate and delete their feedback entries. | |
| Precondition**:** | - The student or denator is logged into the feedback System.  - The student or denator has selected the "Give Feedback" option.  - The appropriate feedback form is available.  - The feedback System is operational. | |
| Post condition: | - The feedback provided by the student or donor is stored in the System.  - The user receives a confirmation message for feedback submission, modification, or deletion. | |
| Assumption: | - The user is successfully logged into the System.  - The feedback System can handle various types of feedback.  - The feedback form includes validation and error handling.  - Users have the ability to modify or delete their feedback if needed.  - Feedback comments, ratings, and details are accurately stored. | |

Figure 4.17 : Give Feedback Use Case Analysis

|  |  |  |
| --- | --- | --- |
| Use case Name: | Chat | |
| Actor(s): | Student, Denator, Admin, Moderator | |
| Description: | This use case allows students, denators and admins to engage in a chat conversation with a moderator for assistance or support. | |
| Typical Course of Events: | Actor Action | System Response |
|  | **Step 1:** student or denator navigates to the "Chat with Moderator" section.  **Step 3:** student or denator initiates a chat request and provides a brief description of their issue or query.  **Step 5:** A moderator accepts the chat request.  **Step 6:** (student or denator) and moderator engage in the chat conversation to address the issue or query | **Step2:** The System provides access to a chat interface.  **Step 4:** The System notifies the available moderators of the chat request.  **Step 7:** The System facilitates the chat conversation and stores the chat history.  **Step 8:** The System allows student or denator to end the chat session and provides a summary of the conversation. |
| Alternate Courses: | - Moderator Unavailable: The student or donor initiates a chat request, The System notifies the moderators, but all are currently unavailable, The System informs the student or donor that no moderator is currently available and suggests alternative contact methods. | |
| Precondition**:** | - The users are logged into the System.  - Moderators are available to respond to chat requests. | |
| Post condition: | - The chat conversation is recorded and stored for reference.  - The user's issue or query is addressed. | |
| Assumption: | - The users are successfully logged into the System.  - Moderators are available to handle chat requests.  - The System can facilitate real-time chat communication.  - The chat conversation is recorded for quality and reference purposes. | |

Figure 4.18 : Chat Use Case Analysis

### : Activity Diagram:

* An Activity Diagram is used to describe the sequential flow of activities of a use case (flow of functionality) in a system. An Activity Diagram is a flow chart showing the flow of activities through the system.

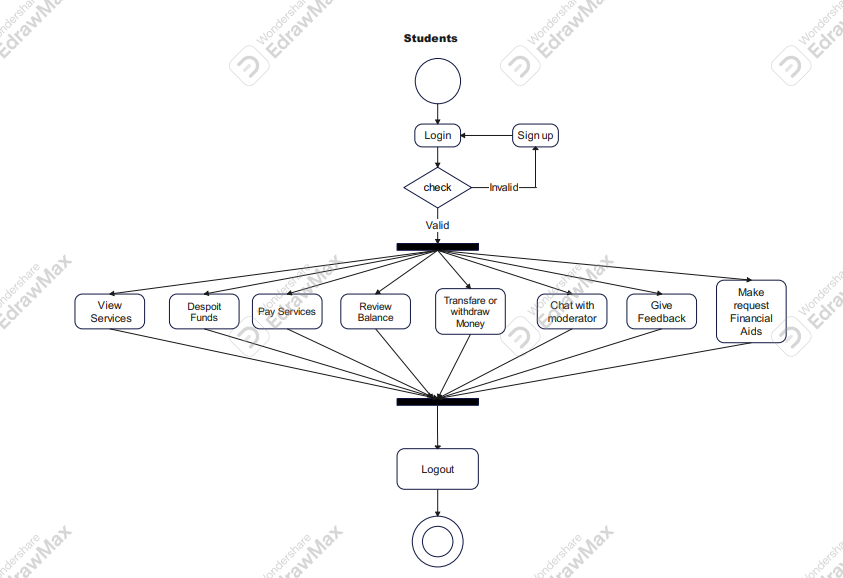


Figure 4.19 : Student activity Diagram

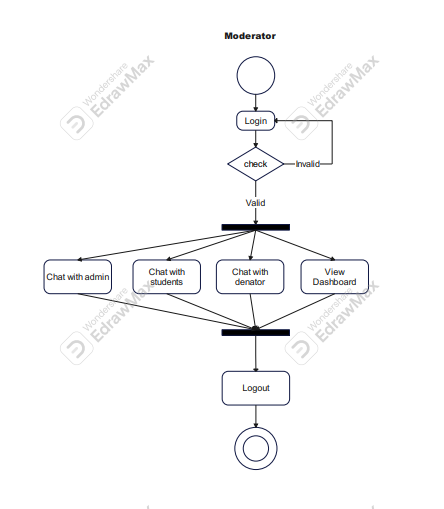


Figure 4.20 : Moderator activity Diagram

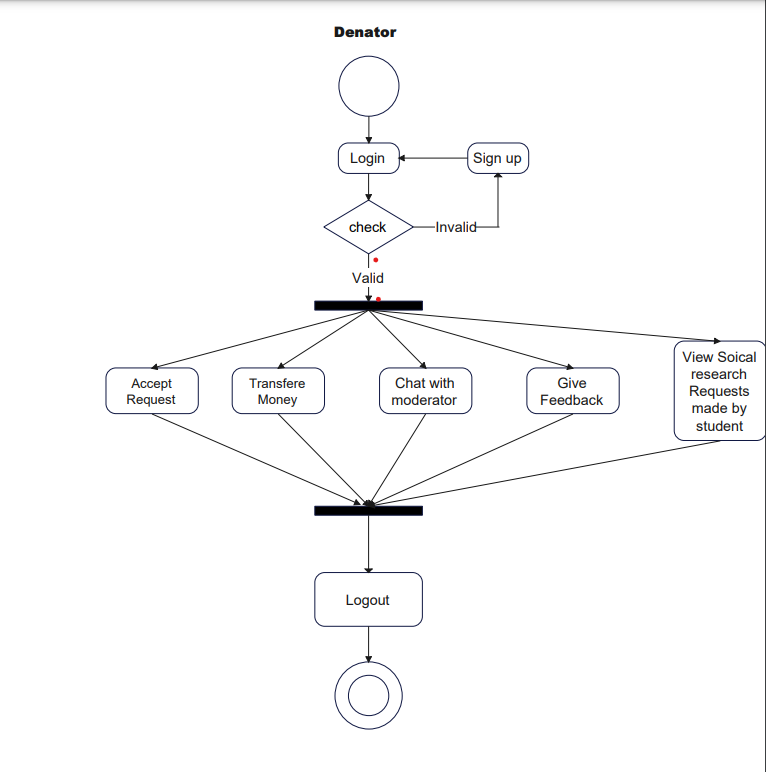


Figure 4.21 : Denator activity Diagram

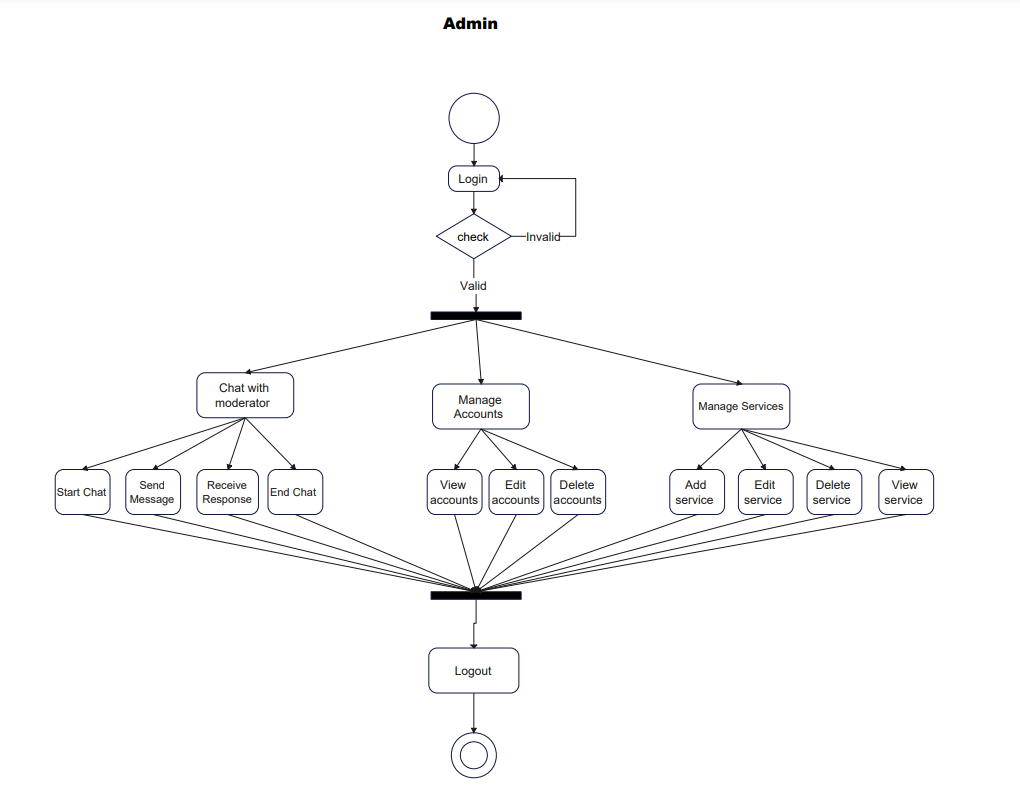


Figure 4.21 : Admin activity Diagram

### Non-Functional Requirements A non-functional requirement (NFR) :

* is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are often called the "quality attributes" of a system. Non-Functional requirements such as:
  1. **Usability**: refers to the quality of a user's experience when interacting with a system, it appears in the ease of use through Navigation, Purpose of features, and Quality of performance.
  2. **Reliability**: consistency of the system appears in the system being available all the time.
  3. **Scalability**: the ability of the system to handle the growth of data due to demand - e.g., the number of students registering simultaneously.
  4. **Performance**: Refers to how well the software system accomplishes certain functions under specific conditions - e.g. Sending the registration form to the advisor and the confirmation back to the student quickly enough.
  5. **Portability**: Defines how well the system elements may be accessed and may interact from two different platforms - e.g. A student can start the registration process from the website and finish it on the mobile application.
  6. **Security**: protect sensitive data against unauthorized access - e.g. only authorized users can access students’ data.

****

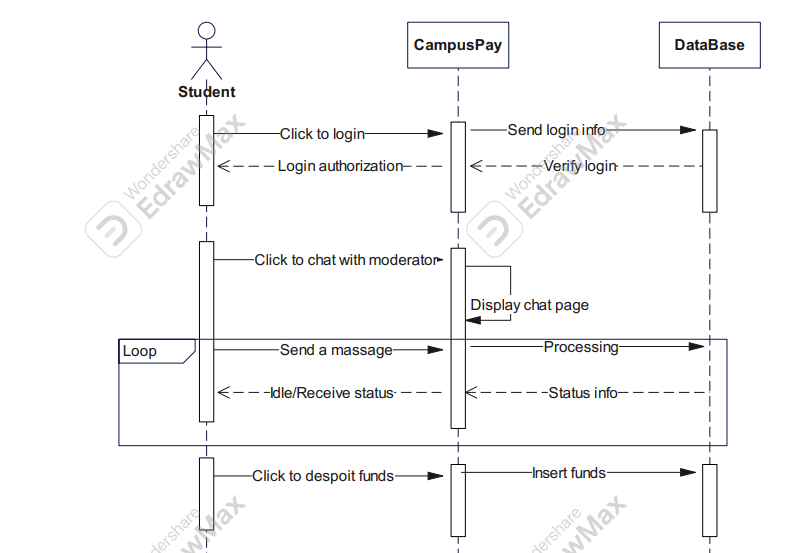
**Chapter 5**

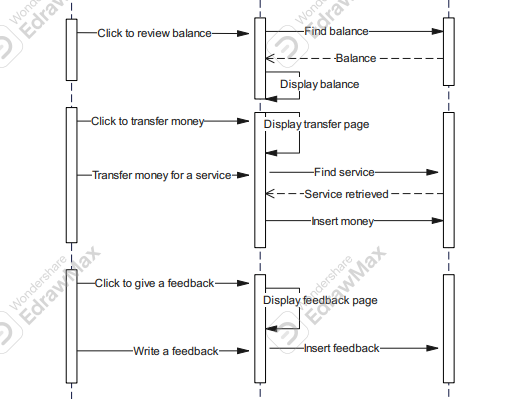
**System Design**

**4.1 Overview :**

* This chapter describes how the system works by discussing System Sequence diagrams, Sequence diagrams, class diagram, ERD and includes a framework.
* A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur, Class Diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects, Context Diagram which presents the sub-systems of our system and its data flow processing, and system architecture is the conceptual model that defines the structure, behavior, and more views of a system.

**4.2 System Sequence Diagram A system sequence diagram (SSD)**

* is a sequence diagram that shows, for a particular scenario of a use case, the events that external actors generate their order, and possible inter-system events



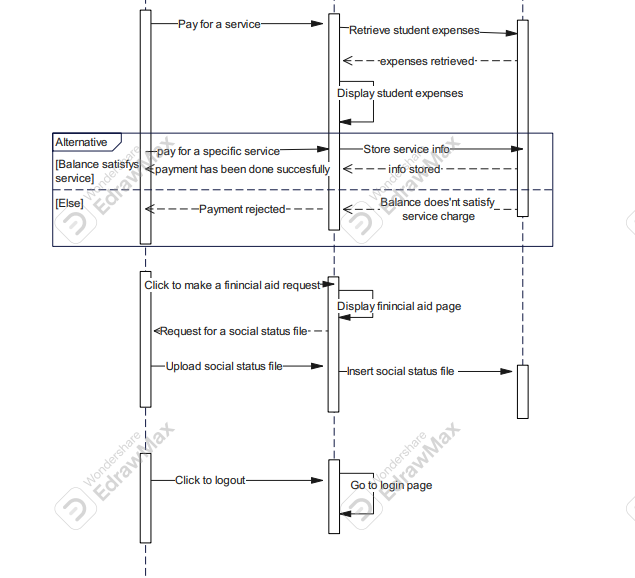
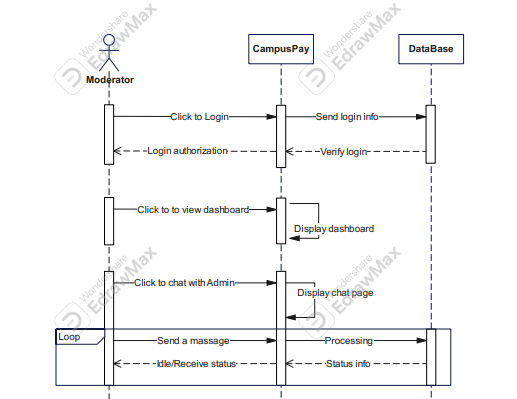


Figure 5.1 : Student Sequence Diagram



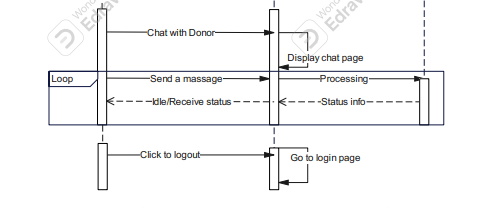
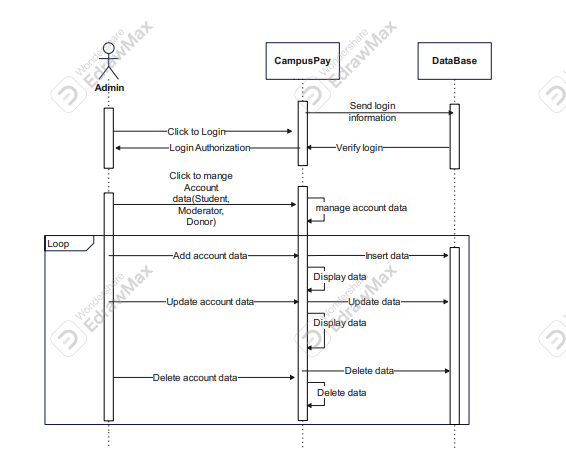


Figure 5.2 : Moderator Sequence Diagram



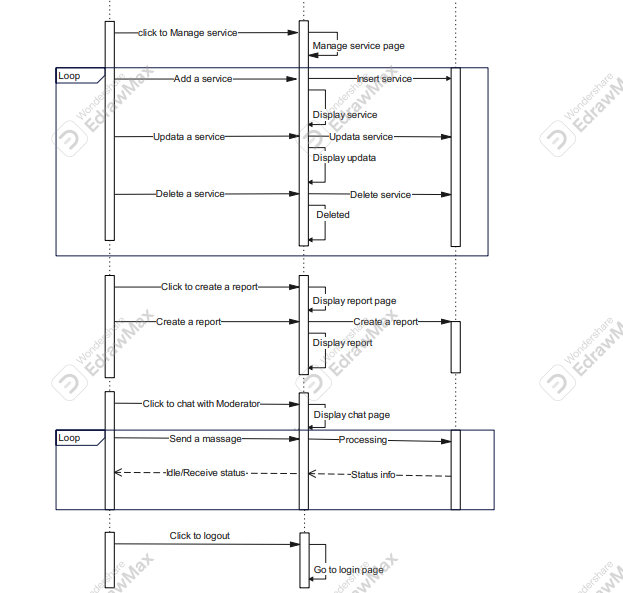
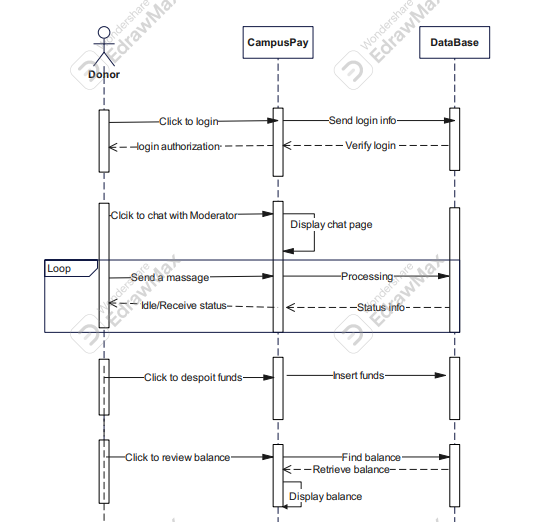


Figure 5.3 : Admin Sequence Diagram



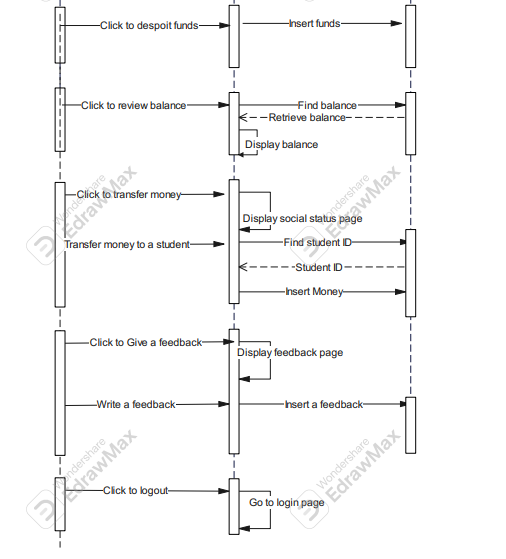


Figure 5.4 : Denator Sequence Diagram

**4.3 Class Diagram :**

* A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

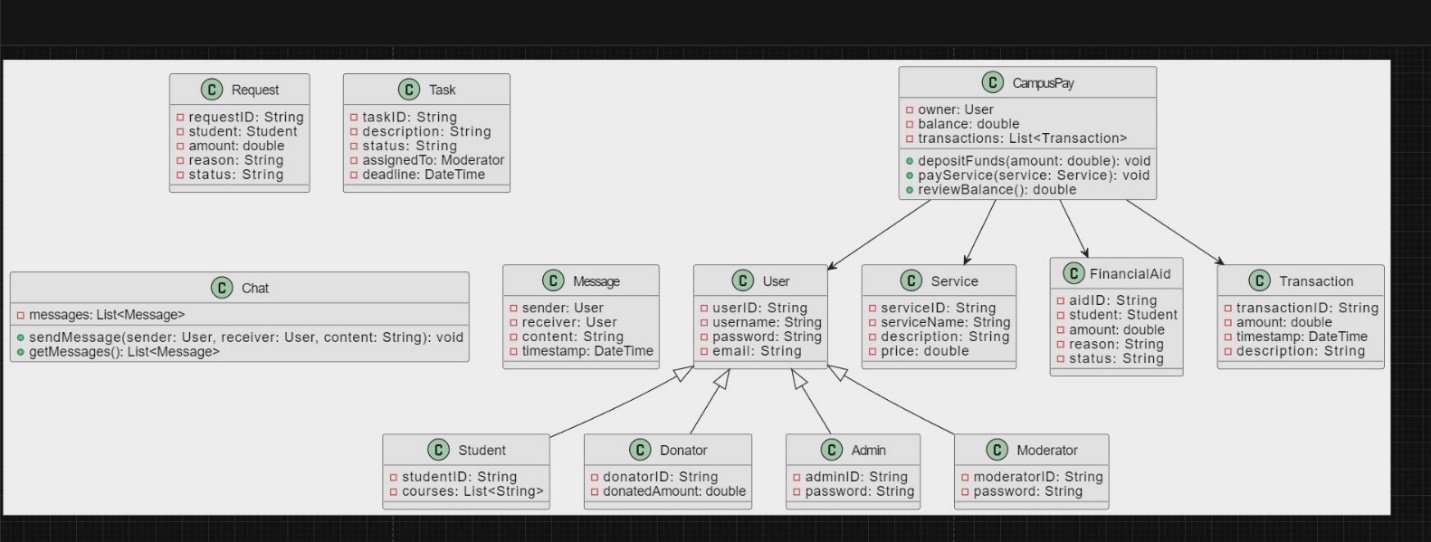


Figure 5.4 : Class Diagram

**4.4 Data Modeling :**

**4.4.1 Relational Database Diagram (ERD) :**

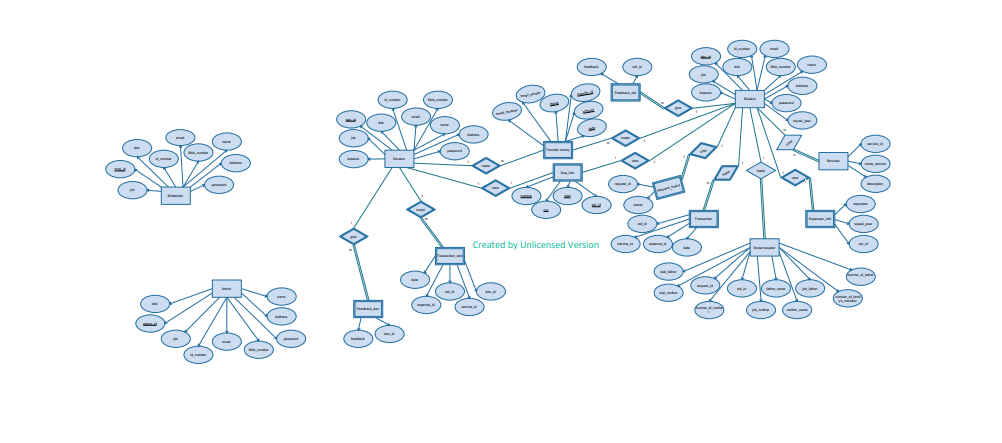
* ****An entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion.

Figure 5.5: ERD Diagram

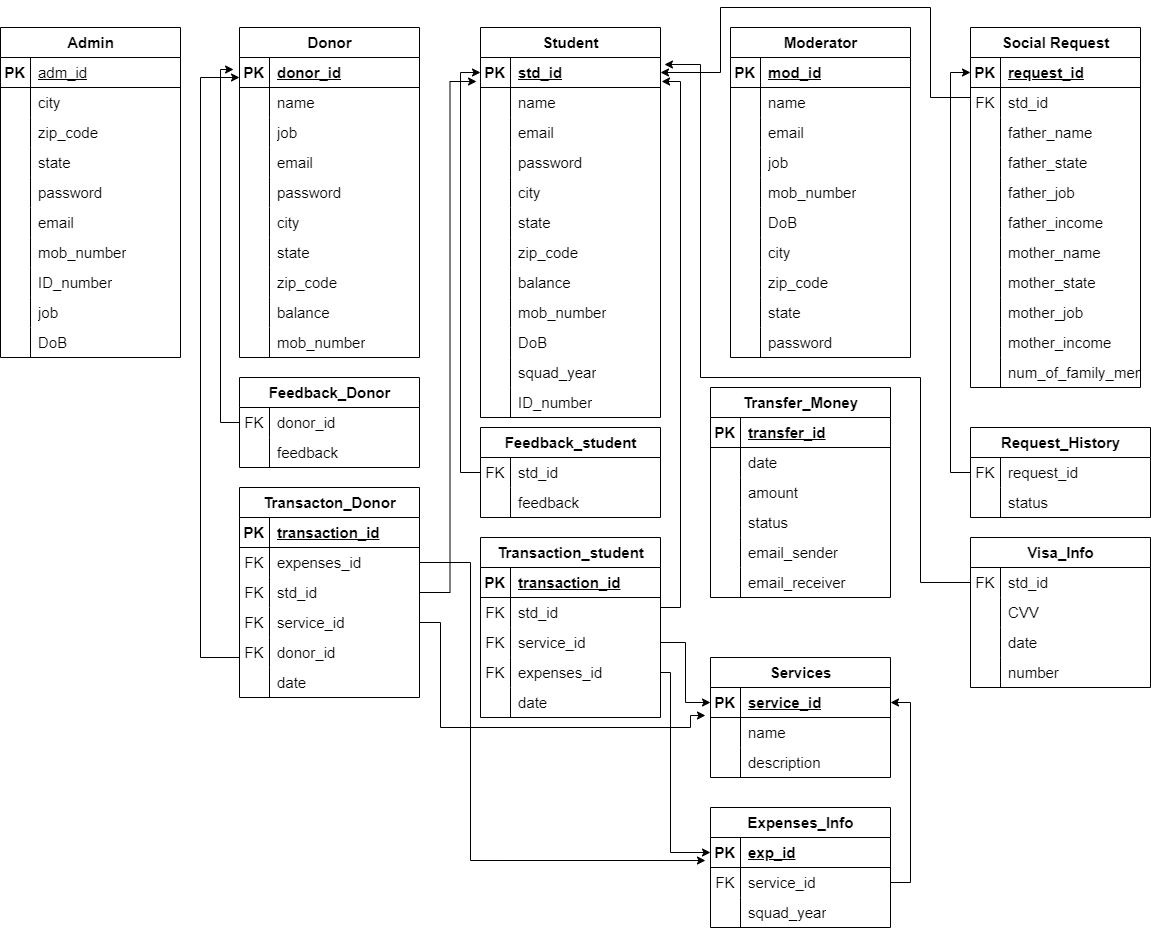


Figure 5.6: Information model Diagram

****

**Chapter 6**

**System Development**