PRINTSTATION
PROJECT REPORT

SUBMITTED TO THE

THE SCHOOL OF ENGINEERING AND INFORMATION TECHNOLOGY

IN THE PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE

COMPLETION OF COURSE

PROJECT WORK IN INDUSTRY/UNIVERSITY/ R&D ORGANIZATION

(CSE499)

SEMESTER VIII

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

BY

AVISHKAR B. KOLAHALU - 1322029

S. SREYA RATNIKA - 1322014

HENISHA DINESH SURU - 1322005

UNDER THE GUIDANCE AND SUPERVISION OF

DR. SUKHWANT KAUR

MANIPAL UNIVERSITY
DUBAI CAMPUS

SCHOOL OF ENGINEERING AND INFORMATION TECHNOLOGY

MANIPAL UNIVERSITY DUBAI CAMPUS

DUBAI, UNITED ARAB EMIRATES

(JUNE 2017)

CERTIFICATE

This is to certify that the Report Titled

PrintStation

Describes the bona fide work done by

AVISHKAR B. KOLAHALU - 1322029
S. SREYA RATNIKA - 1322014
HENISHA DINESH SURU - 1322005

For the completion of the course
PROJECT WORK IN INDUSTRY/UNIVERSITY/ R&D ORGANIZATION (CSE 499)
Offered in the Eighth Semester of the Academic Program

Bachelor of Technology
In
Computer Science & Engineering

During the Academic Year
2016 – 2017

_____
Faculty Guide

_____
Internal Examiner

_____
External Examiner

External Exam held on: _____

DECLARATION

We hereby declare that the matter embodied in this Graduation Project work entitled is the result of the analysis and research work carried out by us under the guidance of **Dr. Sukhwant Kaur** School of Engineering and Information Technology, Manipal University Dubai campus, U.A.E. This work has not previously been submitted with, associates or any other similar title, to any candidate of any university.

AVISHKAR B. KOLAHALU - 1322029
S. SREYA RATNIKA - 1322014
HENISHA DINESH SURU - 1322005

B. Tech Computer Science & Engineering,
Manipal University, Dubai campus,
United Arab Emirates.

ACKNOWLEDGEMENT

Project work in university has been a great learning and self-development opportunity for us as students. We consider ourselves very lucky and honored to have so many wonderful people lead us through the completion of this project.

We are grateful to our Project Guide, Dr. Sukhwant Kaur for her willingness to give us valuable advice and direction whenever we approached her with a problem. We are thankful to her for providing immense guidance and continuous encouragement for the completion of this project.

We would also like to thank our Project Coordinator and all the faculty members for their constant support throughout the project. Their interest and inputs in our project has been very motivating.

AVISHKAR B. KOLAHALU - 1322029
S. SREYA RATNIKA - 1322014
HENISHA DINESH SURU - 1322005

B. Tech Computer Science & Engineering,
Manipal University, Dubai campus,
United Arab Emirates.

Table of Contents

List of Figures

List of Tables

Abstract

There is an inherent problem with the current way of availing services from university stationeries. Students and faculty would have to be mindful of when the stationery is open and being operated by the person(s) in charge. Queues and long wait times are another point of concern, especially at peak times. Students & faculty would have classes and other tasks that require their attention and cannot afford to spend too much time at the stationery. Not to mention the hassle of obtaining files to be printed from different USBs, Hard drives, and even e-mail.

What we intend to do with our product, **PrintStation**, is to ameliorate the hassles of the printing process at universities. It adds value to the services provided by the university to its students and faculty. PrintStation will consist of two parts: a web portal and a mobile application. The system grants students and faculty the ability to seamlessly upload their required document(s) and allow them to schedule prints from anywhere, through the internet. They may also place orders for stationery, such as pens or files, which would be ready for them when they go to collect it. PrintStation makes the process much easier for the employees in charge of printing as well. Instead of the tedious process of collecting USBs or Hard drives, and sorting through e-mail to obtain the files for print, they would easily be able to view all the orders through the system on the web portal, and files that require printing.

The faculty and students will greatly benefit from this product. They will have access to a reliable service to print their documents and place orders for stationery. This service eliminates the requirement of physical presence at the stationery to get print jobs done, and it provides an easier access for purchasing stationery. The administrator would also find it much easier to print documents through the portal, i.e. one hub where all the files to print are available. It is a queued & sorted process, benefitting greatly from time and resource management.

Keywords: PrintStation, web portal, printing, stationary, online

1. Introduction

1.1. Problem Statement

Students waste a lot of time in queues at the university photocopy centre waiting to print out documents or buy supplies. The current method of walking to the stationary, handing the document in a USB which might contain a virus and then searching the USB for the document is time consuming and inefficient. The student then must give instructions to the admin to print out the document.

Faculty also have difficulty in printing out the required paperwork as it might not be ready in time for the next time. So, the faculty could instead upload a file the night before and the print would be ready the next morning.

1.2. Proposed System

PrintStation will be a dedicated web portal and an android application for students and faculty to submit their print requests and get their documents printed. The system will allow students and faculty to log on with their account, upload the document(s) they want to print and then send their print request from anywhere that has internet access. Both, students & faculty, can also place orders for stationery supplies such as pens, files, etc. They can then go to the stationary and collect their printouts & supplies without the hassle of waiting in queues, providing USBs or Hard Drives containing the file(s), and then waiting for printing job to complete. The administrator at the stationery will also have a much easier task of printing out said documents, since they are all in a standardized format and are all available in one place.

The purpose of this system is to make the printing, binding and purchasing process at a university more convenient and efficient. The system would work based on a credit system and benefit both the students and the university.

The system will streamline and standardize the process of printing & shopping for supplies. The service would be provided to students as a credit based service where students can preload their account with credits to print or buy supplies. The credit system would not apply to the faculty. This service eliminates the requirement of physical presence at the stationery to get print jobs done, and it provides an easier access for purchasing stationery.

1.3. Stakeholders

The stakeholders of the PrintStation will be:

- The developers
- The Project mentor who guided the team with the entire development of the system.
- The administrator at the Photocopy Centre who will oversee getting the prints ready and making sure that the shop is updated periodically.
- The students and the faculty of the university who will be the end users for the application.

1.4. Functional Requirements

1.4.1. Login

The system should allow users to enter the official registration number and password provided by the university as their username and password to login. The system then checks to see if the username and password are valid credentials by comparing them to the combination held in the server database. If the credentials are valid then the system should take the user to the respective dashboard. If the credentials are invalid, then an error message should be shown and the user is asked to re-enter the credentials.

The user must be allowed to change their password if desired.

1.4.2. Faculty

The system must identify and allow the user as faculty. The system must ensure that the user has access only to the faculty dashboard. The faculty should be able to seamlessly upload the documents they need for print. They should also be able to make an order for the stationary with ease. It must be ensured that the faculty do not pay for their purchases.

1.4.3. Student

The system must identify and allow the user as student. It must ensure that the user has access only to the student dashboard. The students should be able to seamlessly upload the documents they need for print. They should also be able to make an order for the stationary with ease.

The students must not be able to pay the order when they have insufficient number of credits.

1.4.4. Admin

The administrator resides in the stationary room and carries out the print request. The system identifies the user as admin. Once he receives a request he can download the file(s), print them out and keep it ready for the user. The administrator would also be able to update the stock quantity of supplies on the application, which would be useful information for the students and faculty.

1.5. Non-Functional Requirements

1.5.1. Performance Requirements

PrintStation would be used within the university which interacts with the student and the faculty. It must be ensured that the database performs all the required functionalities. The performance of the system should be fast and accurate. The system should be able to handle the large amounts of data uploaded by the users. Maximum allotted response time for a change to occur should be 5 seconds. Response time for database should be 6 seconds.

1.5.2. Security Requirements

Security is one of the most important factors. Only authorized users will be able to access the system. The students and the faculty would be provided with their login details to the emails registered with the university. Every user must be given access only to the functionalities they are given authorized to. Only the admin will have the rights to modify the databases, the user should not have any access to them.

1.5.3. Reliability

The server should be reliable so that it does not crash and accurately transmits information to the database. In case of any technical problems, it should be able to retrieve the data lost through a backup that needs to be regularly maintained and updated to reflect the most recent changes.

1.5.4. Availability

The system should be available always meaning the user can access it using a web browser at any time of the day, only restricted by the down time of the server on which the system runs. Maintenance should be scheduled many days in advance so that the faculty can avoid any clash between the test timings and the maintenance timings.

1.6. Main Features

- **User friendly interface:** The proposed system is user friendly because the retrieval and storing of data is fast and data is maintained efficiently. Moreover, the graphical user interface is provided in the proposed system, which provides user to deal with the system very easily.
- **Credit-based payment:** This system will have introduced the concept of credits in the systems. The payments will be made using credits instead of by cash manually.
- **Notifications:** The users will get notified when their orders are ready to go pick them up from the photocopy centre.
- **No more queues:** The users need not queue at the photocopy centre to get their print jobs done. They can come and collect once the admin notifies on completion of the print job.
- **Free credit for new users:** 100 credits will be available on registration to every new user in the university.
- **Incentives:** The users will receive bonus credits depending on the number of credits you have spent.

1.7. Hardware Requirements

Web application:

The system can run ASP.NET scripts on a web browser, along with SQL server support for database connectivity. The other hardware requirements are stated below -

- Processor – Intel Pentium 4 or higher.
- Processor speed – 1.5 GHz or higher.
- Operating system – Windows 7 or higher / OS X Mavericks 10.9 or higher.
- RAM – 512 MB or higher.
- Memory requirements – Minimum 60 GB.

Mobile Device:

- Processor – Qualcomm Snapdragon 810 or higher.
- Operating system – Android OS, v7 or higher.
- RAM – 2GB or higher
- Memory requirements – Minimum 32 GB.

1.8. Software Requirements

The software requirements for the project are:

Operating System – Windows 7 or higher

- IDE – Microsoft Visual Studio 2017
- Front End – ASP.net, HTML & CSS; Xamarin Framework
- Language – C#.NET, Java
- Database – MS SQL server, SQLite

2. Methodology

2.1. Waterfall Model

PrintStation is developed using Waterfall Model. The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Implementation and Maintenance. The Waterfall Model is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. At the end of each phase, a review takes place to determine if the project is on the right path or if there is any changes to make the project. In waterfall model, phases do not overlap.

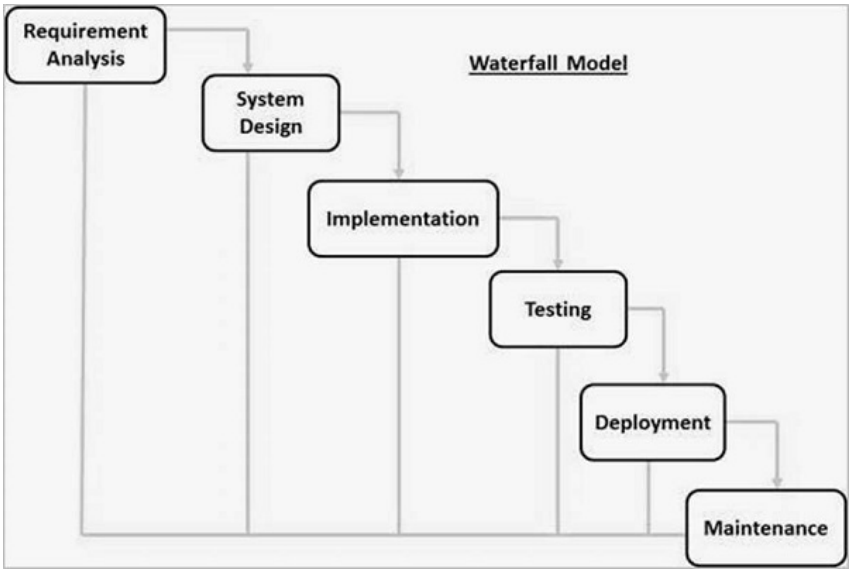

Figure 2.1: Phases of Waterfall Model

2.1.1. Phases in Waterfall Model

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and helps in defining overall system architecture.

- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment. [1]

2.2. Technologies Used

2.2.1. ASP.NET

ASP.NET is a set of Web development tools offered by Microsoft. Programs like Visual Studio .NET and Visual Web Developer allow Web developers to create dynamic websites using a visual interface. Of course, programmers can write their own code and scripts and incorporate it into ASP.NET websites as well. Though it often seen as a successor to Microsoft's ASP programming technology, ASP.NET also supports Visual Basic.NET, JScript .NET and open-source languages like Python and Perl.

ASP.NET is built on the .NET framework, which provides an application program interface (API) for software programmers. The .NET development tools can be used to create applications for both the Windows operating system and the Web.

Programs like Visual Studio .NET provide a visual interface for developers to create their applications, which makes .NET a reasonable choice for designing Web-based interfaces as well. For an ASP.NET website to function correctly, it must be published to a Web server that supports ASP.NET applications. Microsoft's Internet Information Services (IIS) Web server is by far the most common platform for ASP.NET websites. While there are some open-source options available for Linux-based systems, these alternatives often provide less than full support for ASP.NET applications.

Benefits of using ASP.NET:

- **Compiled Code:** ASP.Net provides greatly increased performance by running compiled versus interpreted code.
- **Language Support**: While ASP supported VBScript (a subset of Visual Basic) and Jscript (a subset of Java), ASP.NET supports standard programming languages including Visual Basic, C# and C++.
- **Strict Coding Requirements:** Developers are forced to adhere to strict coding standards. This promotes a professional engineering approach to ASP.NET application development and results in code that is generally less buggy than its ASP counterparts.
- **Event-Driven Programming Model:** All ASP.NET objects on a Web page expose events that can be processed by ASP.NET code. Handling events such as Load, Click and Change via code reduces program complexity and increases organization.
- **Third-Party Controls**: The ASP.NET community has been growing rapidly for the last two years. There is a large base of high-quality third-party controls that can be obtained to significantly speed up development time and reduce cost.
- **User Authentication:** ASP.NET supports forms-based user authentication, including cookie management and automatic redirection of unauthorized logins. ASP.NET allows for user accounts and roles thus providing a high degree of granularity for controlling access to objects and pages.
- **Easier Configuration & Deployment:** Configuration is done with plain text files which can be uploaded or changed while the application is running. There is no need to restart the web server to affect a change.
- **Object and Page Caching**: ASP.NET objects and pages can be cached to dramatically increase performance. The caching system is very advanced, allowing you to specify what needs to be cached and what doesn't and when to recall the cached information rather than perform a new request.
- **Higher Scalability**: ASP.NET has some great scalability features built into it, including maintaining session state across servers, and multi-processor load balancing. [2]

2.2.2. Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, WindowsForms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source control systems and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle.

These are some of the various applications that can be built using Visual Studio:

- **Console applications**: These applications run from the command line and do not include a graphical interface, but are great for small tools or anything that will be run by another application.
- **Windows forms applications**: These are Windows desktop applications written using the .NET framework; since they are .NET applications, they require that the .NET framework be on any computer that will run the application.
- **Windows services**: Services are applications that run in the background while your computer is running. These are usually applications that must perform scheduled tasks or handle continuous network requests.
- **ASP.NET applications**: ASP.NET is a powerful technology that is used to create dynamic web applications, often driven by a database. Many popular websites are written using ASP.NET, including those of e-commerce giants like Dell.
- **ASP.NET web services**: ASP.NET provides a complete web services model that allows you to quickly and easily create web services.
- **Windows Mobile applications**: Windows Mobile applications can run on devices that include the Compact framework; these include Pocket PC devices, as well as cell phones running the Microsoft Smartphone platform.

- **MFC/ATL/Win32 applications**: You can also still create traditional MFC, ATL, or Win32 applications using C++. These applications do not need the .NET runtime to run, but also don't include many of the benefits of working with the .NET framework.
- **Visual Studio add-ins**: That's right, you can use Visual Studio to write new functionality to be added into Visual Studio.
- **And more**: Visual Studio also includes projects to deploy your application, work with databases, create reports, and more.

The following are the features of Visual Studio:

- **IntelliSense**: IntelliSense is the trademark feature of Visual Studio. IntelliSense simply helps you while programming by showing you the available classes and the methods and properties available on those classes. Can't remember what the name of that class, method, or property is? No worries, IntelliSense will help.
- **Designers**: Visual Studio includes visual WSYIWYG designers for Windows applications, ASP.NET applications, and Windows Mobile applications. These designers make it much easier to get your application looking just right.
- **Debugging**: One of the most important features of Visual Studio is the ability to step through your application line by line as it is executing. Not sure why you are getting an error? Simply walk through and see exactly what is going wrong.
- **Organization**: Visual Studio is built for developing applications, so it provides intuitive methods for organizing your various code files into projects and your various projects into solutions.

2.2.3. Xamarin Studio

Xamarin Studio, the Integrated Development Environment (IDE) used to create iOS, Mac and Android applications, presenting the functionality that makes it an excellent tool for creating native mobile and desktop applications on the Xamarin platform.

Xamarin Makes suitable for Cross-Platform App Development for the following reasons:

- **Native applications** - The preferred solution for any application is native design. This is where Xamarin and its unique approach come in. Xamarin Studio (IDE) enables code completion in C#. It provides the advantages of native UI, access to specific-device features, and most importantly, native performance. Code sharing across platforms is a breeze with Xamarin, helping you shorten that development cycle.

- **Shared app logic** - Apart from native UI, the way app logic is shared across multiple platforms makes Xamarin a must-use cross-platform development tool. Application logic underlying the UI layer, like input validation, web service calls, database interactions, and back-end enterprise integrations are coded once in C#.
- **API integration** - Xamarin binds the same APIs and UI controls that are used to build iOS, Android and Mac apps in their respective platform specific languages. With Xamarin, support is always around the corner. Their unique binding technology enables them to provide support for new features soon after they are introduced in the device's operating system.
- **Xamarin Component Store** - Developers can choose from a host of free or paid components, which include UI controls, cross-platform libraries, and third party web services to apps with just a few lines of code
- **Advantages of C#** - C# is a simple, modern, general-purpose, type-safe, pure object-oriented programming language. Being object oriented from the get-go, it is more than just a modernized version of C++, allowing even simple data types to be treated as objects. Type-safety of C# helps prevent type errors that could result in flawed program behavior, without the need of boilerplate or verbose type annotations. [3]

2.2.4. SQL Database

SQL stands for Structured Query Language. SQL is used to communicate with a database. Per ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc. Although most database systems use SQL, most of them also have their own additional proprietary extensions that are usually only used on their system. However, the standard SQL commands such as "Select", "Insert", "Update", "Delete", "Create", and "Drop" can be used to accomplish almost everything that one needs to do with a database.

SQL allows us to do the following:

- execute queries against a database
- retrieve data from a database
- insert records in a database

- update records in a database
- delete records from a database
- create new databases
- create new tables in a database
- create stored procedures in a database
- can create views in a database
- set permissions on tables, procedures, and views [4]

3. Models, Attributes and Functions in the System

3.1. Class Diagram

A class diagram in the Unified Modelling 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. The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed. In the diagram, classes are represented with boxes that contain three compartments:

· The top compartment contains the name of the class. It is printed in bold and centred, and the first letter is capitalized.

· The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.

· The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

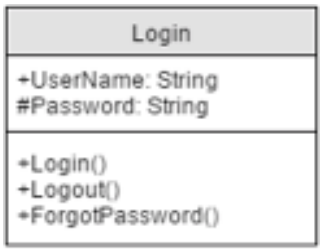

Figure 3.1: Class Diagram Example

To specify the visibility of a class member (i.e. any attribute or method), these notations must be placed before the member's name:

- \+ Public
- \- Private
- # Protected
- ~ Package

A relationship is a general term covering the specific types of logical connections found on class diagrams. The various types of relationships in a class diagram are:

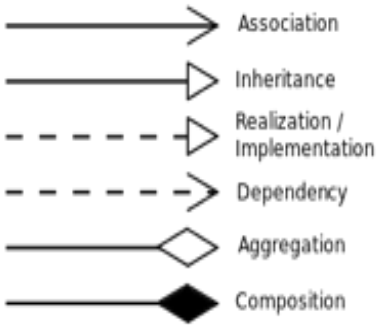

Figure 2.2: Class Diagram Relationships

- **Dependency**

  A dependency is a semantic connection between dependent and independent model elements. It exists between two elements if changes to the definition of one element (the server or target) may cause changes to the other (the client or source). This association is uni-directional.

- **Association**

  An association represents a family of links. A binary association (with two ends) is normally represented as a line. An association can link any number of classes. An association with three links is called a ternary association. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties. There are four different types of association: bi-directional, uni-directional, aggregation (includes composition aggregation) and reflexive. Bi-directional and uni-directional associations are the most common ones.

- **Aggregation**

  Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship. As a type of association, an aggregation can be named and have the same adornments that an association can. However, an aggregation may not involve more than two classes; it must be a binary association. In UML, it is graphically represented as a hollow diamond shape on the containing class with a single line that connects it to the contained class.

- **Generalization**

  It indicates that one of the two related classes (the subclass) is a specialized form of the other (the super type) and the superclass is considered a Generalization of the subclass. The UML graphical representation of a Generalization is a hollow triangle shape on the superclass end of the line (or tree of lines) that connects it to one or more subtypes. The

generalization relationship is also known as the inheritance or "is a" relationship. The superclass (base class) in the generalization relationship is also known as the "parent", superclass, base class, or base type. The subtype in the specialization relationship is also known as the "child", subclass, derived class, derived type, inheriting class, or inheriting type.

- **Realization**

  In UML modelling, a realization relationship is a relationship between two model elements, in which one model element (the client) realizes (implements or executes) the behaviour that the other model element (the supplier) specifies. The UML graphical representation of a Realization is a hollow triangle shape on the interface end of the dashed line (or tree of lines) that connects it to one or more implementers. A plain arrow head is used on the interface end of the dashed line that connects it to its users. A realization is a relationship between classes, interfaces, components, and packages that connects a client element with a supplier element. (Ambler, 2009)

- **Multiplicity**

  This association relationship indicates that (at least) one of the two related classes refer to the other. The UML representation of an association is a line connecting the two associated classes. At each end of the line there is optional notation. The various multiplicities are:

  - 0..1 No instances, or one instance
  - 1 Exactly one instance
  - 0..* Zero or more instances
  - 1..* One or more instances [5]

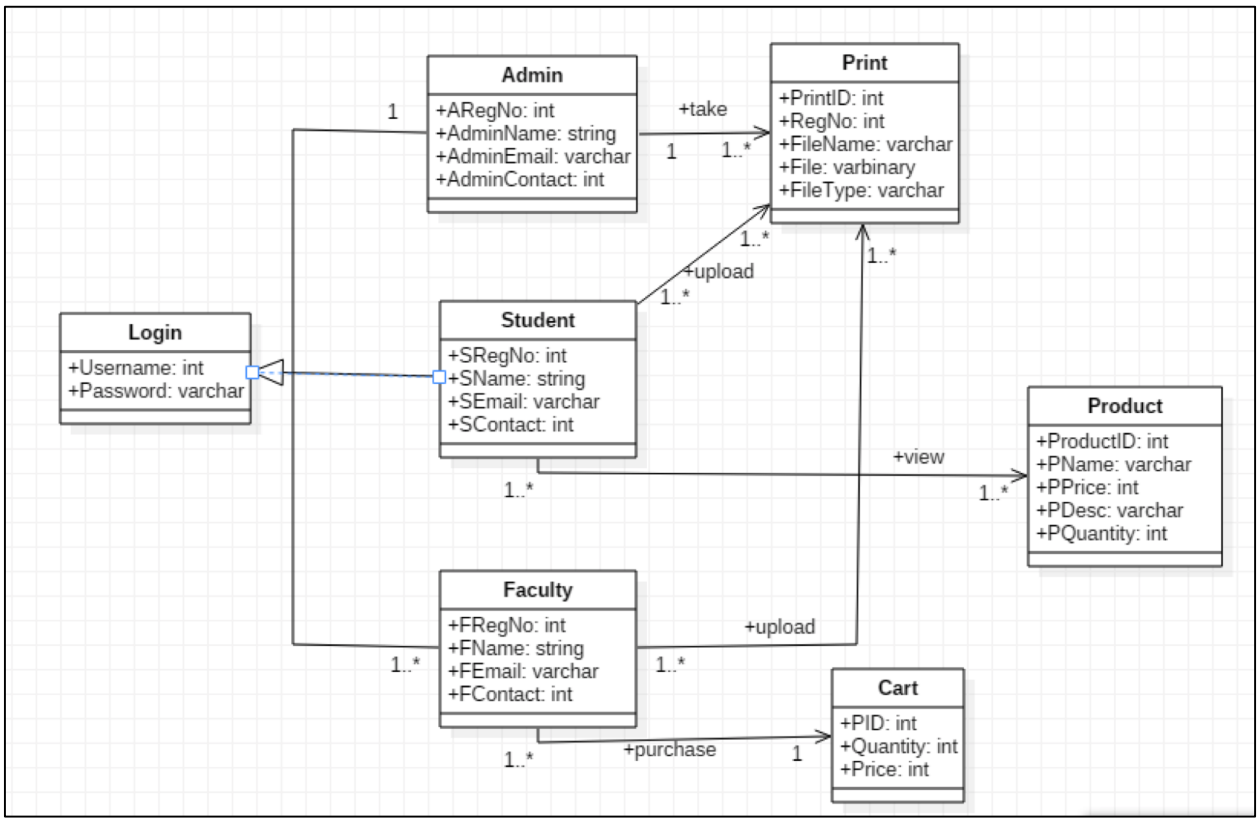

Figure 3.3: Class Diagram for PrintStation

3.2. Use Case Diagram

A use case diagram establishes the capability of the system. It can identify the different types of users of a system.

Components of use case diagram:

Actor:

Actor in a use case diagram is any entity that performs a role in one given system. This could be a person, organization or an external system and usually drawn like skeleton shown below.

Figure 3.4: Actor

Use Case:

A use case **represents a function or an action within the system**. Its drawn as an oval and named with the function.

Figure 3.5: Use Case Diagram

System Boundary:

System boundary is used to define the scope of the use case and drawn as a rectangle. This an optional element but useful when your visualizing large systems. For example, you can create all the use cases and then use the system object to define the scope covered by your project. Or you can even use it to show the different areas covered in different releases.

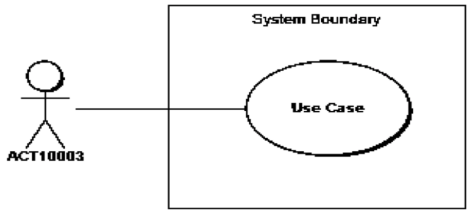

Figure 3.6: System Boundary Diagram

Relationships:

There are five types of relationships in a use case diagram. They are:

- **Association between an actor and a use case**

  An association indicates that an actor can carry out a use case.

- **Generalization of an actor**

  Generalization of an actor means that one actor can inherit the role of another actor. The descendant inherits all the use cases of the ancestor. The descendant has one or more use cases that are specific to that role.

- **Extend relationship between two use cases**

  It extends the base use case and adds more functionality to the system. The extending use case is dependent on the extended (base) use case.

- **Include relationship between two use cases**

  Include relationship show that the behaviour of the included use case is part of the including (base) use case. The main reason for this is to reuse the common actions across multiple use cases. The base use case is incomplete without the included use case.

- **Generalization of a use case**

  This is like the generalization of an actor. The behaviour of the ancestor is inherited by the descendant. This is used when there is common behaviour between two use cases and specialized behaviour specific to each use case. [5]

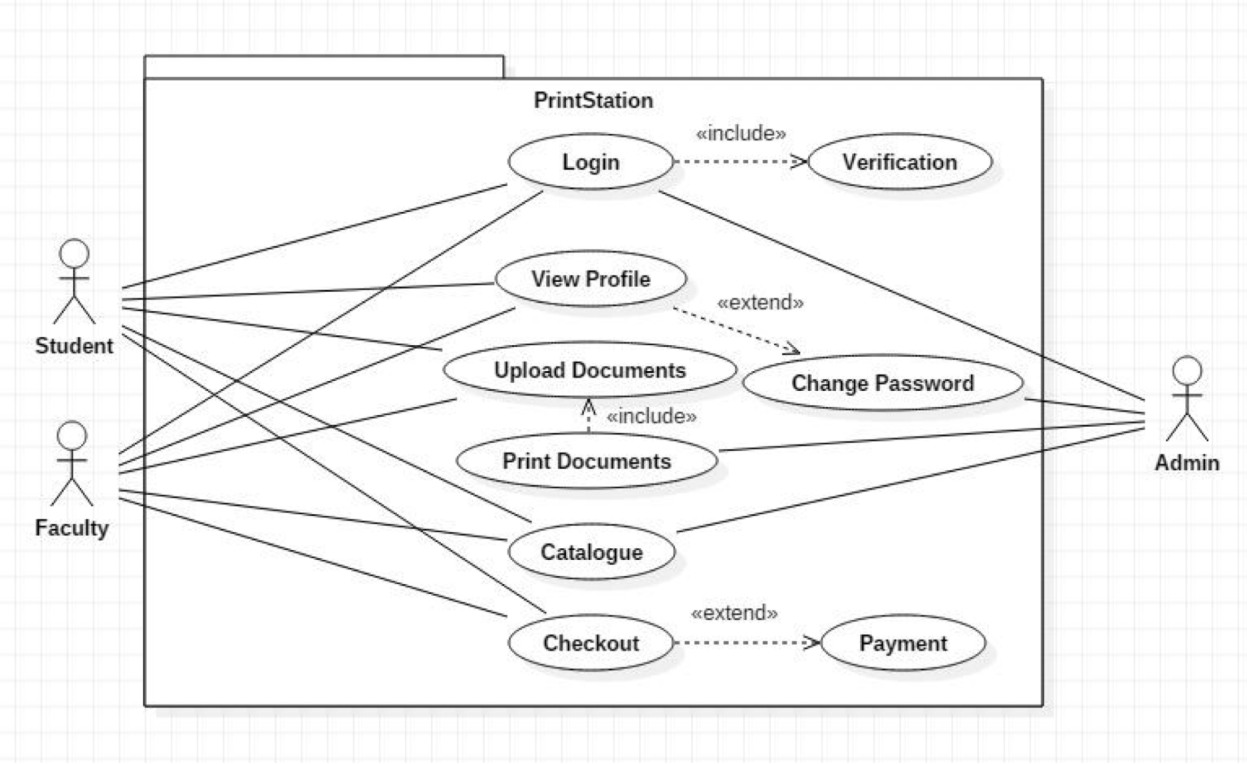

Figure 3.7: Use Case Diagram for PrintStation

For PrintStation we have considered 3 actors and Use Cases.

The Actors are:

- **Admin**

  The administrator carries out the print request. Once the administrator receives a request he can print out the documents and keep it, as well as any stationery supplies, ready for the respective student or faculty.

- **Student**

  The students can use the system to print out various documents that are a part of university work such as assignments, forms, lab records and so on. They can also buy their stationery online using the credit system. Once bought they can go to the stationery and pick them up.

- **Faculty**

  The faculty can also use this service to print out certain documents such as class quizzes, forms, tests and so on. The credit system would not apply to the faculty. The faculty can also order the required stationery using the system and pick them up from the stationery.

The different use cases are:

- **Login**
  - Pre-Condition: Users must have username and password assigned
  - Actors: Student, Faculty, Admin
  - Purpose: To enter valid username and password to get access to the system.

- **View Profile**
  - Pre-Condition: User must login using their username and password.
  - Actors: Student, Faculty, Admin
  - Purpose: They can view their profile details and change password if required.

- **Print Document**
  - Pre-Condition: Admin must login using their username and password.
  - Actors: Admin
  - Purpose: To print out various documents that are a part of university work such as assignments, forms, lab records and so on which are uploaded by the users.

- **Upload Documents**
  - Pre-Condition: User must login using their username and password.
  - Actors: Student, Faculty
  - Purpose: To allows the upload the documents they want to print

- **Catalogue**
  - Pre-Condition: User must login using their username and password.
  - Actors: Student, Faculty, Admin
  - Purpose: The student and faculty can purchase the stationary supplies. The admin can update the catalogue to accommodate the stock of the supplies.

- **Checkout**
  - Pre-Condition: User must have selected the items wish to buy
  - Actors: Student, Faculty

- Purpose: To show a list of items the user had bought and pay for the same.

- **Payment**
  - Pre-Condition: User must have selected the items wish to buy
  - Actors: Student
  - Purpose: The student can use the credits they must make payments.

3.3. Block Diagram

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. Block diagrams are typically used for higher level, less detailed descriptions that are intended to clarify overall concepts without concern for the details of implementation. Contrast this with the schematic diagrams and layout_diagrams used in electrical engineering, which show the implementation details of electrical components and physical construction.

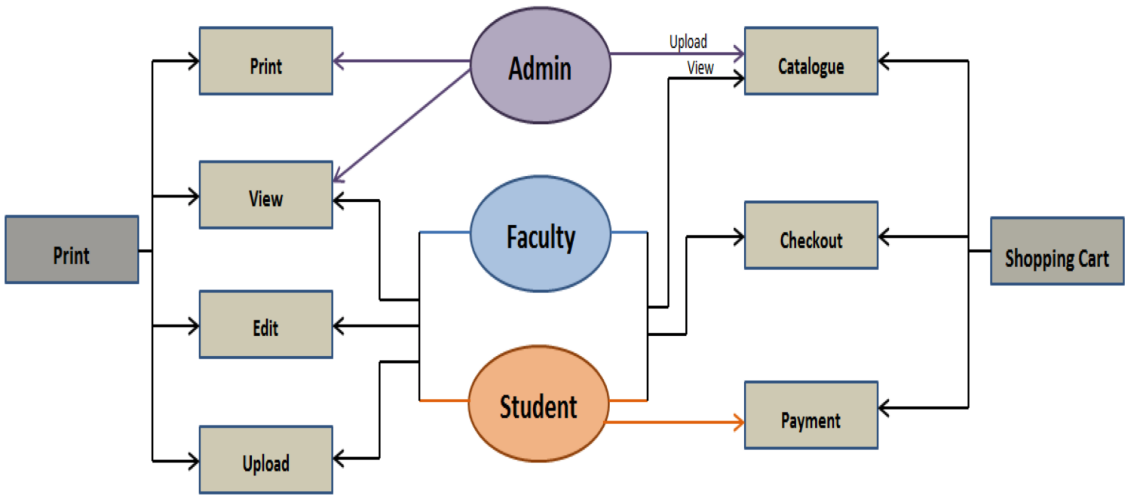

Figure 3.8: Block Diagram for PrintStation

Description of Block Diagram for PrintStation:

The above diagram shows that there are two main parts for the application - the printing and the shopping cart. There are three main users for the system - the admin, faculty and the student. The print part of the application allows faculty and admin to upload, edit and view documents which they want to get printed. The admin can view the documents uploaded by the users and print them. The shopping part of the application allows the faculty to buy the required stationary. The admin will have the able to view and update the catalogue when there are insufficient stocks. The faculty and the user both must checkout for placing any orders. But the

faculty will not have to pay for any orders they make, while the students will have the make the payments.

3.4. Sitemap

A **site map** (or **sitemap**) is a list of pages of a web site accessible to crawlers or users. It can be either a document in any form used as a planning tool for Web design, or a Web page that lists the pages on a website, typically organized in hierarchical fashion. Sitemaps are used to make relationships between pages and other content components. It shows shape of information space in overview. Sitemaps can demonstrate organization, navigation, and labelling system.

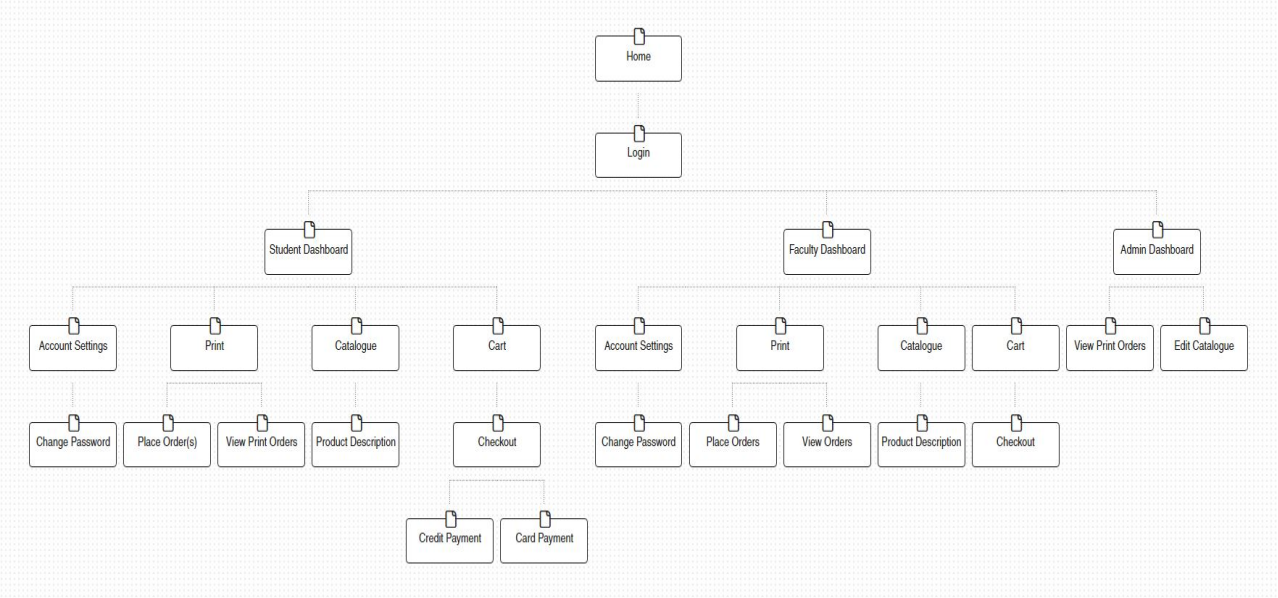

Figure 3.9: Sitemap for PrintStation

The following is the proposed site map for the web application part of PrintStation. It is a list of pages of the web site accessible to users. It clearly shows the pages that each user has access to.

3.5. Database Tables

3.5.1. Login Table

The Login table provides authentication so that only registered users can access the services. The primary key is the Username. This table is going to be used by Students class, Faculty class and the Administrator class.

	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	Username	int	☐	
	Password	varchar(20)	☐	
	Datatype	varchar(20)	☐	

Table 3.1: PrintStation – Database – Login Table

3.5.2. Student Details Table

The student details table helps in providing all the required information about the students. The primary key is ID.

	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	SRegNo	int	☐	
	StudentName	varchar(50)	☐	
	StudentDept	varchar(50)	☐	
	StudentEmail	varchar(50)	☐	
	StudentMobile	int	☐	
	CreditsBal	int	☐	((100))
	CreditSpent	int	☐	((0))

Table 3.2: PrintStation – Database – Student Table

3.5.3. Faculty Details Table

The faculty details table helps in gathering all the required information about the students. The primary key is ID.

	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	FRegNo	int	☐	
	FacultyName	varchar(50)	☐	
	FacultyDept	varchar(50)	☐	
	FacultyEmail	varchar(50)	☐	
	FacultyMobile	int	☐	

Table 3.3: PrintStation – Database – Faculty Table

3.5.4. Admin Details Table

The admin details table will consist of all the general information about the user which is required. The primary key is ID.

	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	ARegNo	int	☐	
	AdminName	varchar(50)	☐	
	AdminEmail	varchar(50)	☐	
	AdminMobile	int	☐	

Table 3.4: PrintStation – Database – Admin Table

3.5.5. Product Table

The product table consists of all the details of every item in the shopping catalogue. The primary key is ID.

	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	RegID	int	☐	
	ProductName	varchar(50)	☐	
	ProductPrice	int	☐	
	ProductQuantity	int	☐	((1))
	Status	varchar(10)	☐	('In Cart')
	Time	date	☐	(getdate())
	TotalCost	int	☐	((0))

Table 3.5: PrintStation – Database – Product Table

3.5.6. Prints Table

The print table would show all the details of the files which are uploaded for printing. PrintID is the primary key.

	Name	Data Type	Allow Nulls	Default
🔑	PrintID	int	☐	
	RegNo	int	☐	
	FileName	varchar(MAX)	☐	
	File	varbinary(MAX)	☐	
	FileType	varchar(MAX)	☐	
	Price	int	☐	
	Status	varchar(10)	☐	('In Cart')
	Time	date	☐	(getdate())
	Colour	varchar(30)	☐	('Black & White')
	Sides	varchar(20)	☐	('Single Sided')
	Binding	varchar(20)	☐	('No Binding')

Table 3.6: PrintStation – Database – Print Table

4. Implementation

4.1. Web Application

The web application was developed using Microsoft Visual Studio 2017, with the debugging and testing services provided by the IDE itself. The database used with the web application was created within Visual Studio, using Microsoft SQL Database and IIS Express. The design/front-end of the web application was done by creating elements using asp.NET and HTML, and the styling of these elements was done using Cascading Style Sheets (CSS). To animate and implement behaviour functionality unto these elements, inline JavaScript was used. This merely represented the aesthetics of the web application. To get the web application's major functionalities to work, programming was done in the Code Behind, i.e. C# programming. Using C#, connections to the database were established so that SQL operations could be performed. Operations such as:

- Authentication.
- Session control.
- Converting files to binary and uploading them into the database.
- Adding stationery supplies to cart.
- Retrieving cart information for checkout.
- Retrieving user account information.
- Retrieving past transaction information.
- Retrieving all user order information for the admin to complete.
- Sending notifications to users informing them that their orders is ready for collection.

API's used to achieve certain functionalities:

- iTextSharp – a .NET PDF Library used to count the number of pages that exists in a PDF document uploaded by students to calculate the price.
- Microsoft Word Object Library – used to count the number of pages in a .doc, .docx, .rtf document uploaded by students to calculate the price.
- SMTP – Simple Mail Transfer Protocol that is used to send email notifications to users when their order is ready for pickup.

4.2. Android Application

The Android application is for users to place orders on catalogue supplies, after authentication is complete. The Android application was also created in Microsoft Visual Studio 2017 utilising similar benefits of programming the back-end with C#. However, front-end designing and development was done using the Xamarin framework which simplifies design process by using XAML forms to design pages of the application. Back-end operations on the database is done using SQLite, a more limited functionality database. It was preferred to MS SQL, as SQLite could be run outside of Visual Studio.

4.3. Screenshots for Web Application

4.3.1. Home Pages

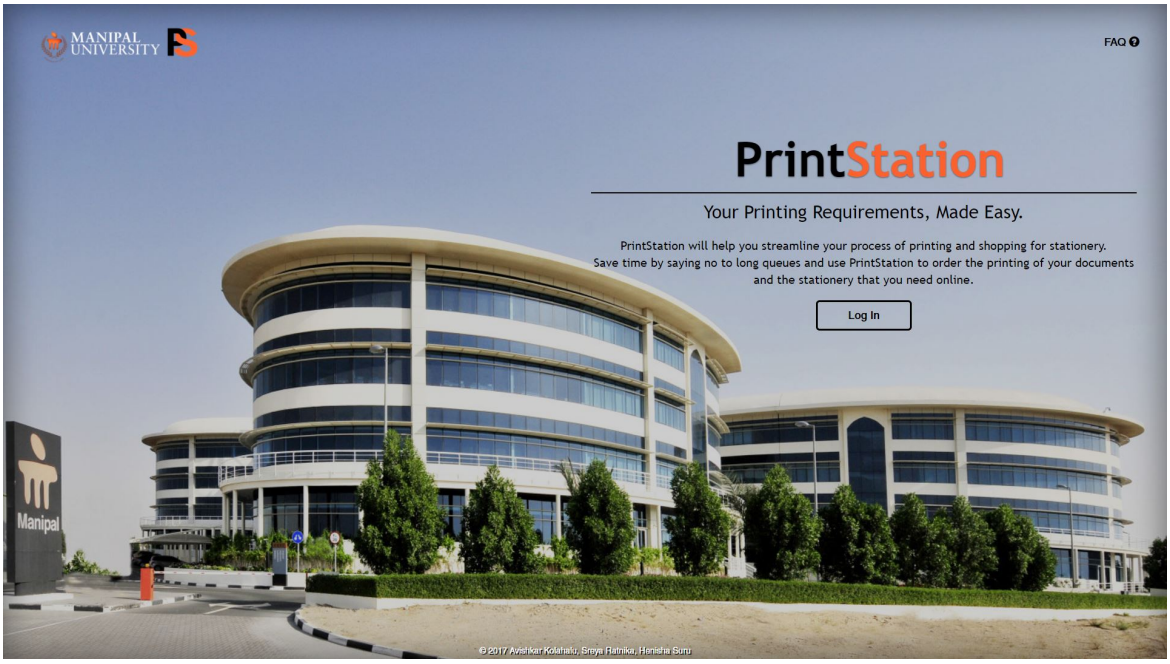

Figure 4.1: PrintStation – Home Page

Figure 4.2: PrintStation – Login Page

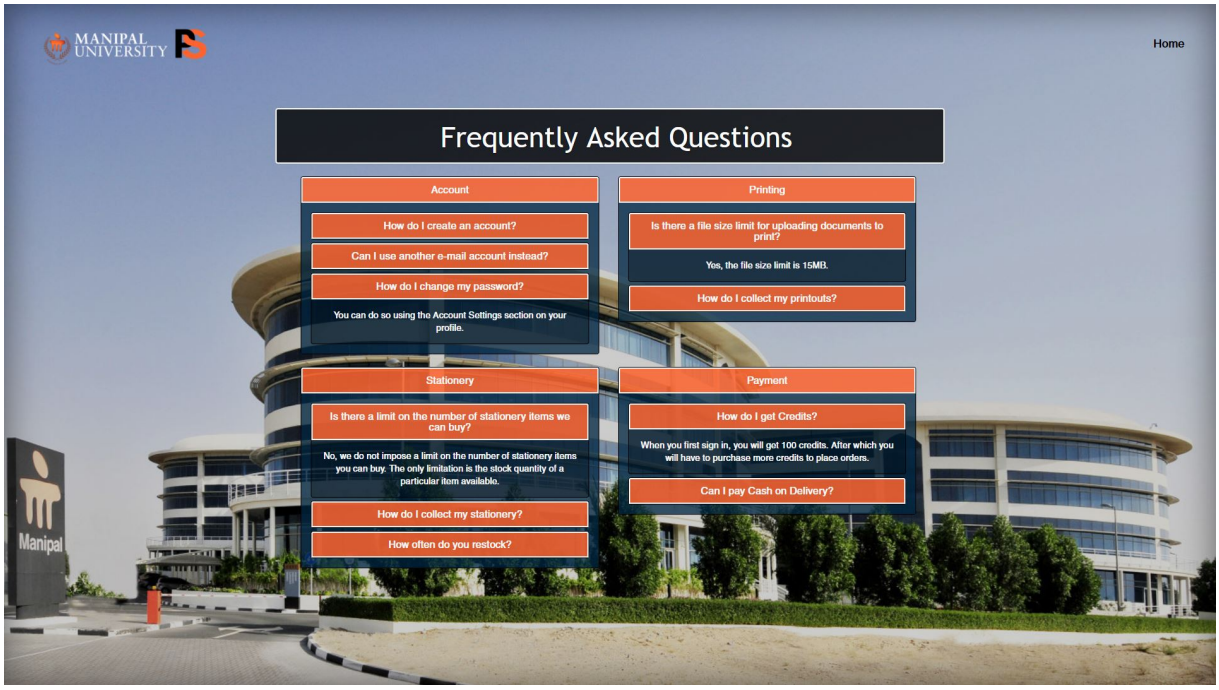

Figure 4.3: PrintStation – FAQ Page

4.3.2. Student Dashboard

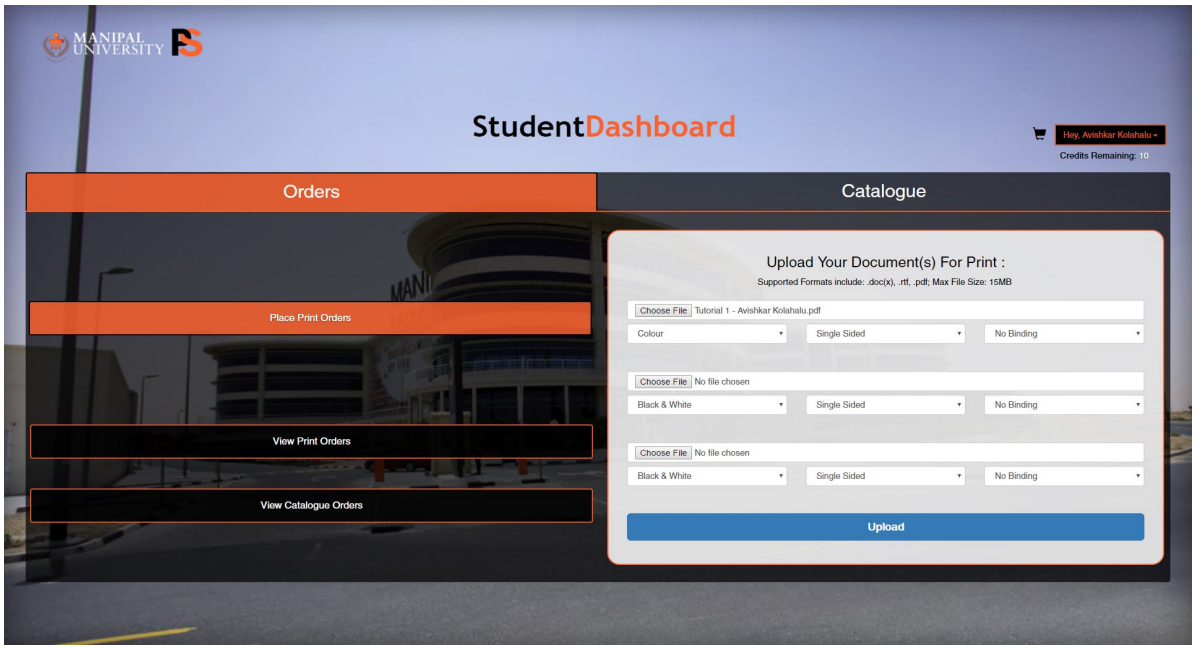

Figure 4.4: PrintStation – Student – Dashboard

Student Dashboard is the first page which opens when student logs in. It consists of two parts- Orders and Catalogue.

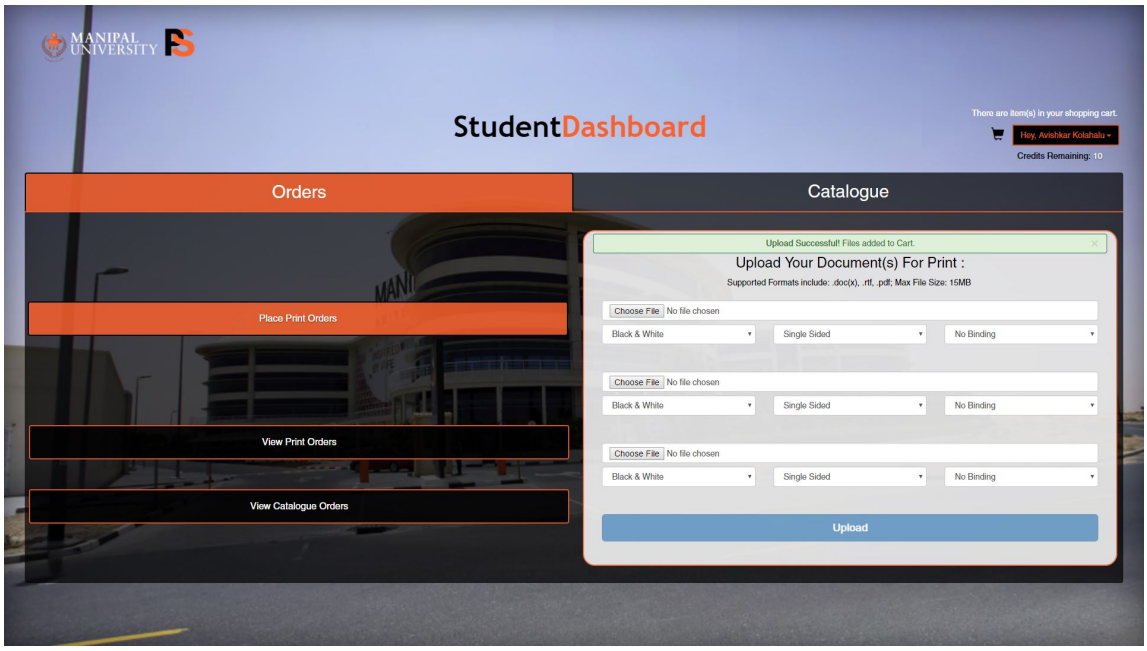

Figure 4.5: PrintStation – Student – Place Print Orders Page

The students can place upload the files they wish to print on this page. They can choose the kind of print they want i.e. if they want it in colour, black and white, single sided, double sided or if they want it to be bound.

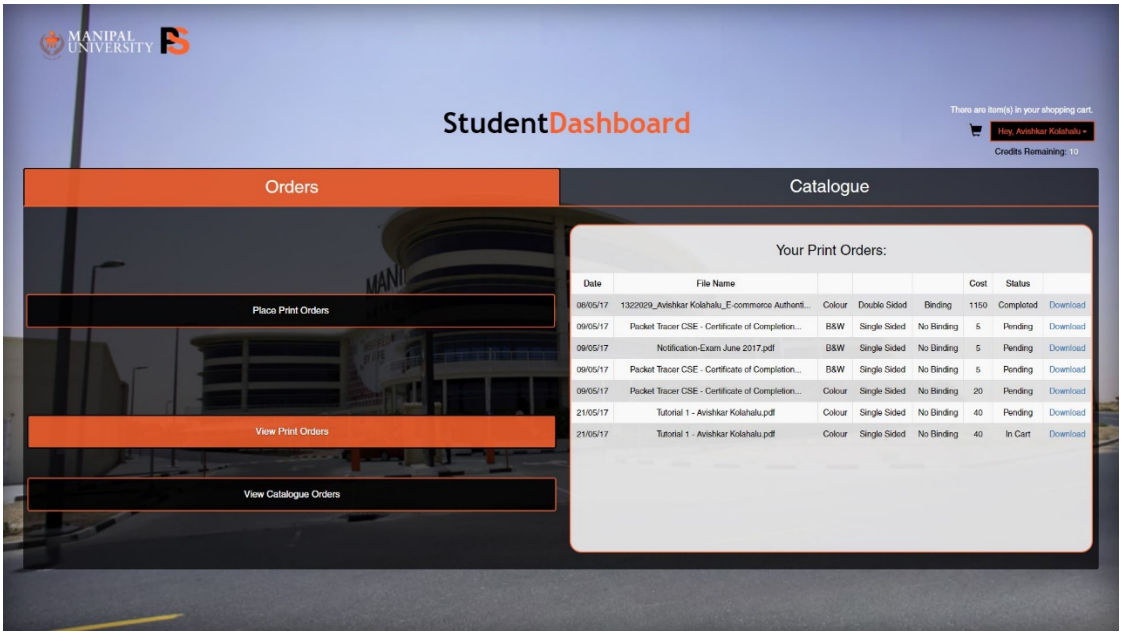

Figure 4.6: PrintStation – Student – View Print Order Page

This page shows that the print orders the student has given so far. It shows the status for every order made.

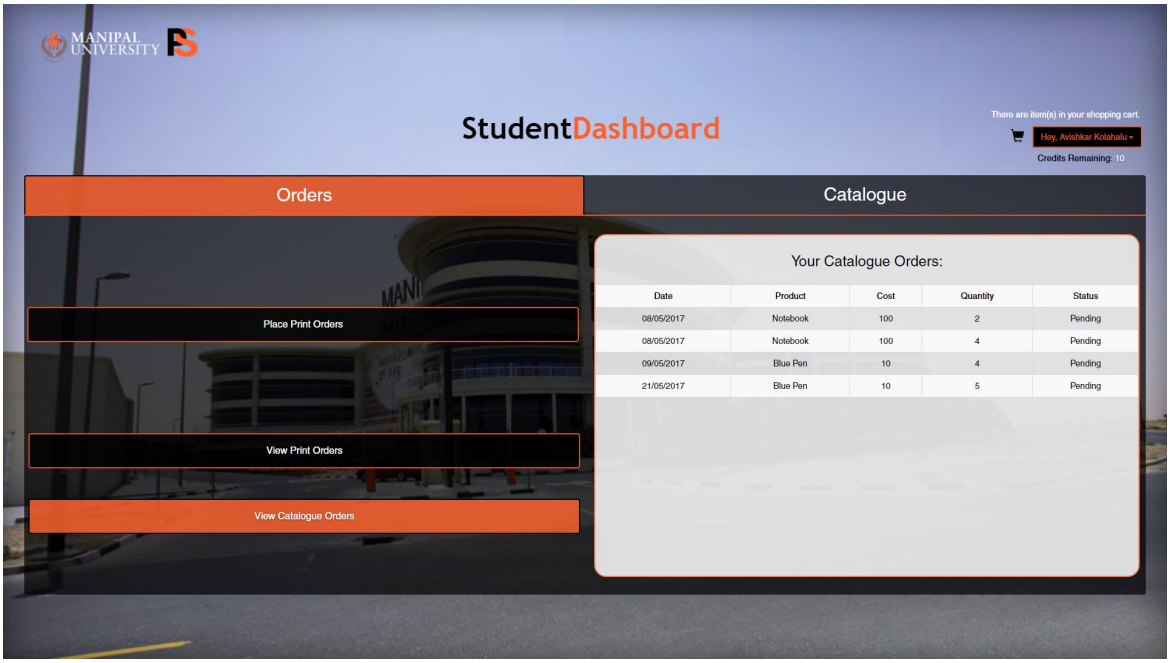

Figure 4.7: PrintStation – Student – View Catalogue Orders Page

This page shows that the shop orders the student has given so far. It shows the status for every order made along with the cost and quantity.

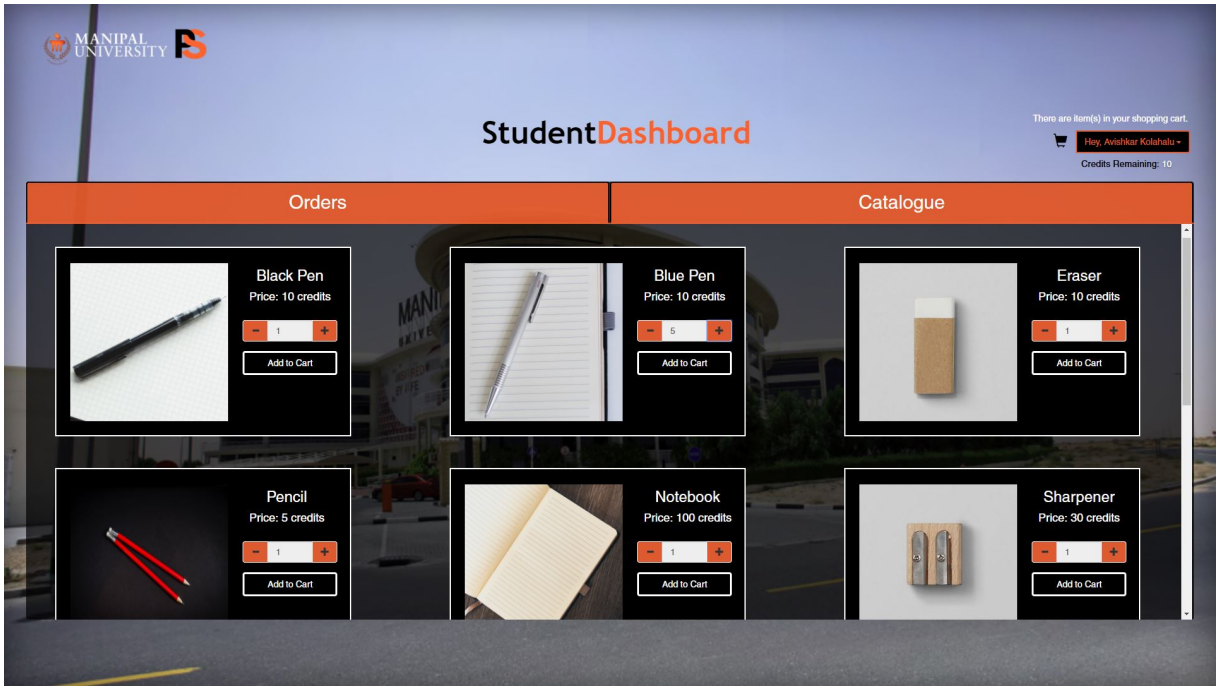

Figure 4.8: PrintStation – Student – Catalogue Page

This page shows the shop catalogue for the students to buy what they require.

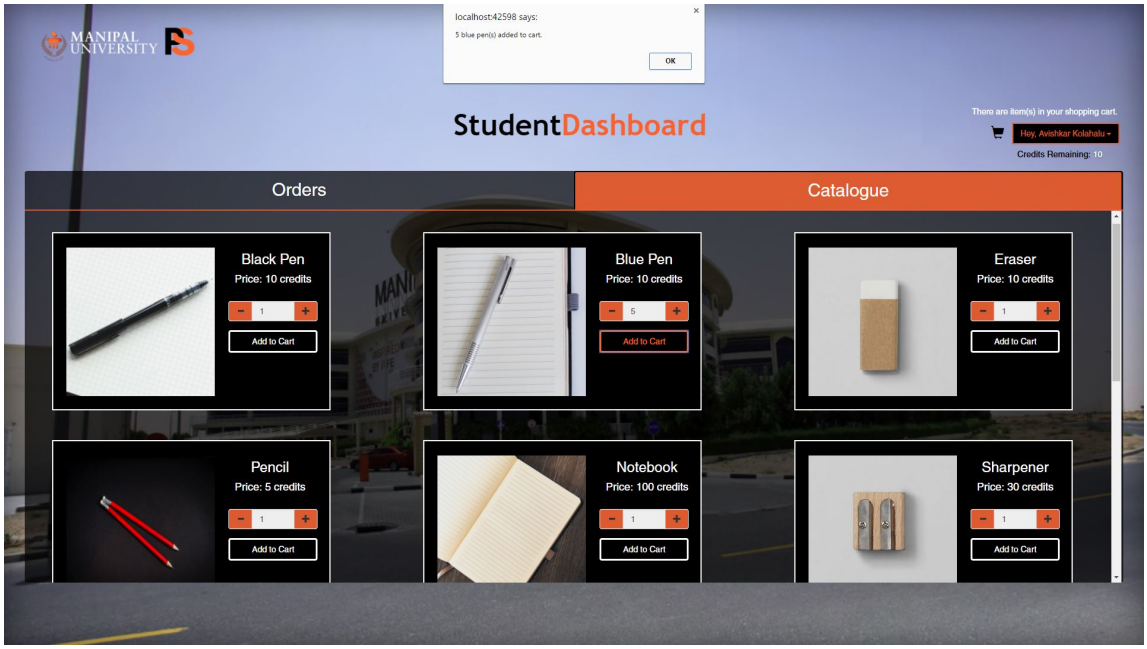

Figure 4.9: PrintStation – Student – Items Added to Cart

This page show an alert which appears when the items are added to the shopping cart.

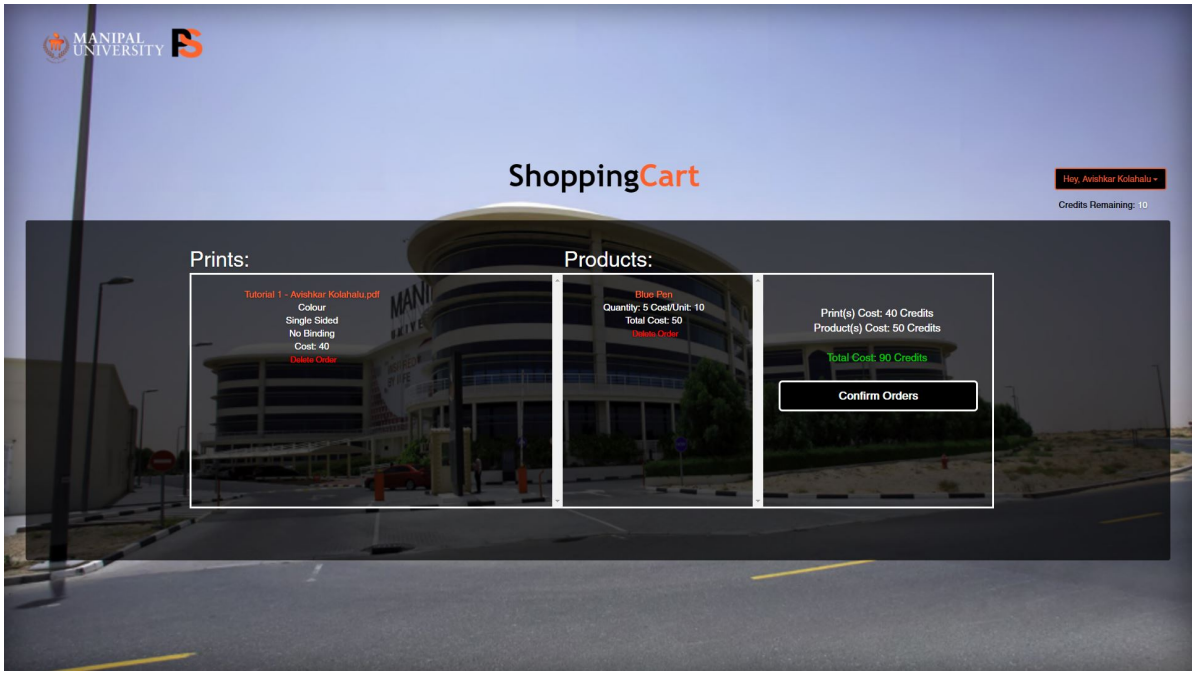

Figure 4.10: PrintStation – Student – Shopping Cart Page

This page shows all the items added to the shopping cart. It shows all the items with the quantity and price, before order confirmation.

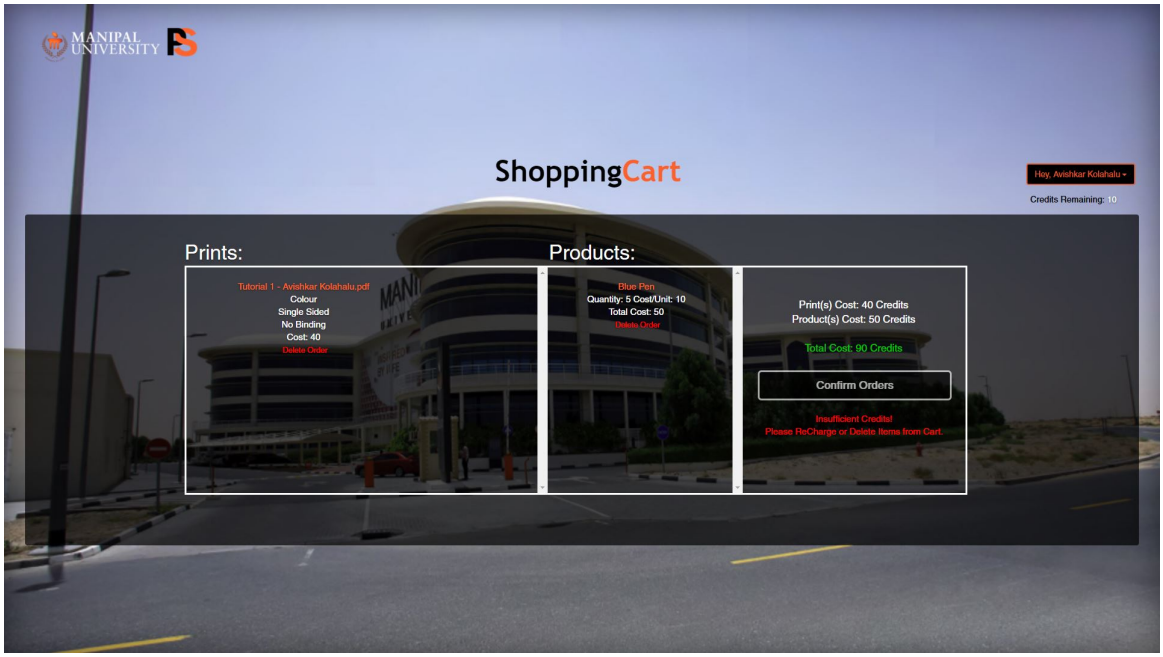

Figure 4.11: PrintStation – Student – Cart – Insufficient funds

An alert stating insufficient funds is show when there aren't enough credits to pay for an order being made.

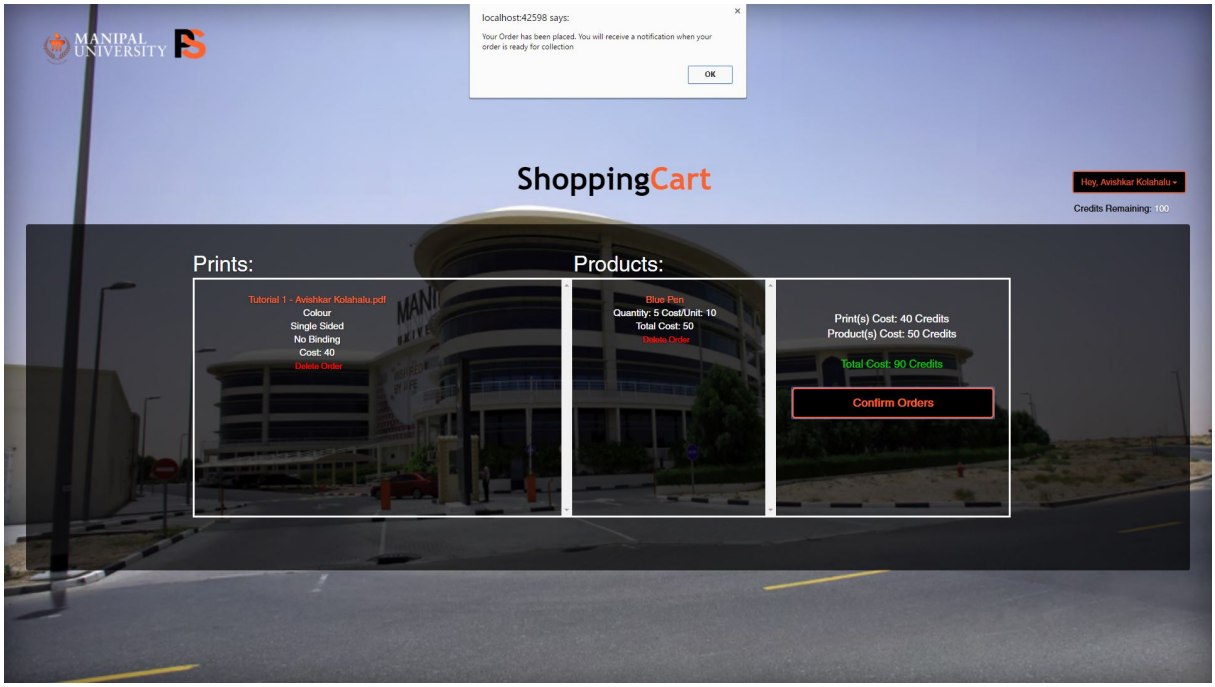

Figure 4.12: PrintStation – Student – Cart Checkout

Once the order is made an alert is shown stating the order has been made successfully.

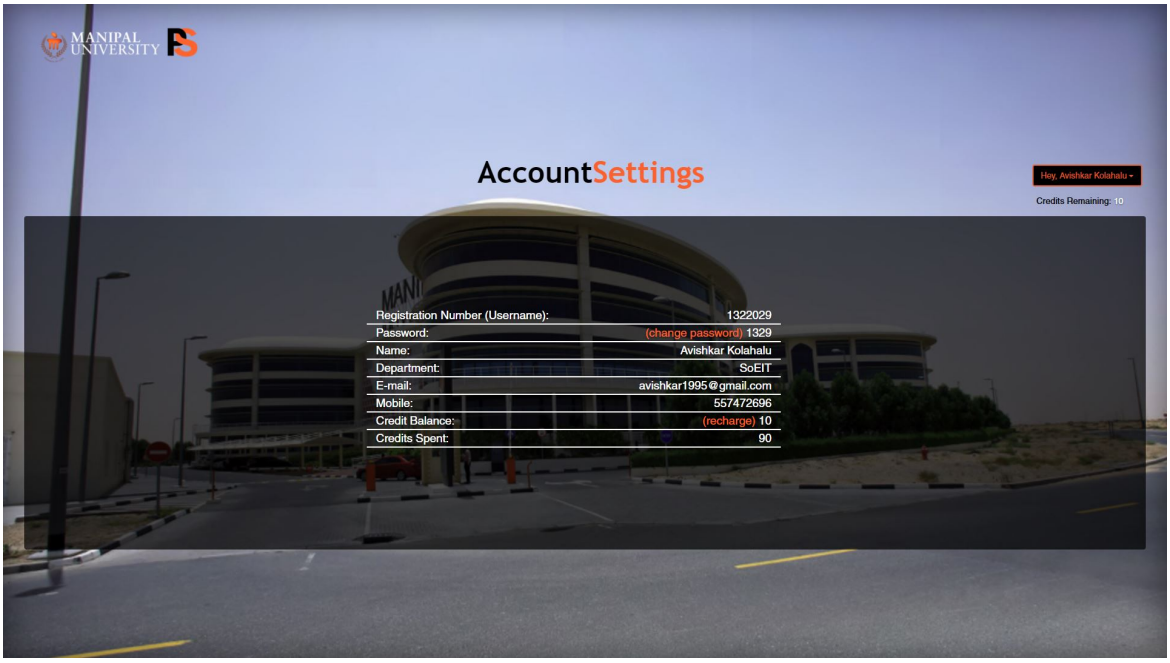

Figure 4.13: PrintStation – Student – Account Setting Page

This page shows the details of user and credit balance. The user can change password on this page.

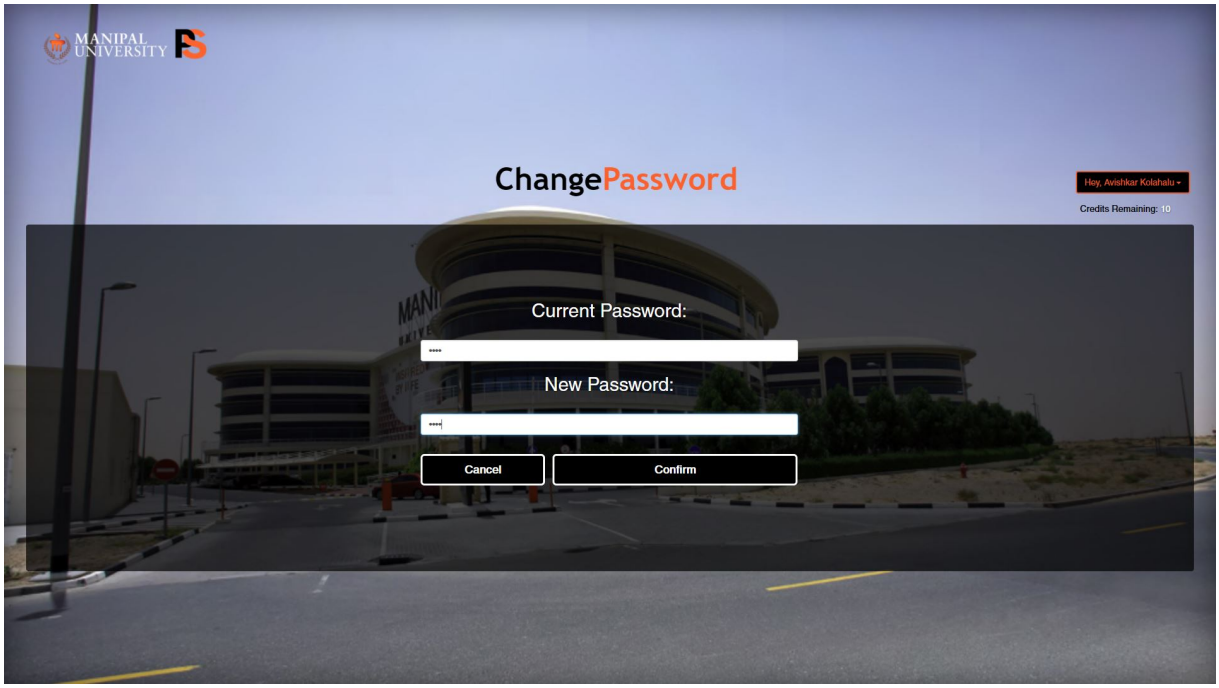

Figure 4.14: PrintStation – Student – Change Password Page

This page allows the user to create a new password for their account.

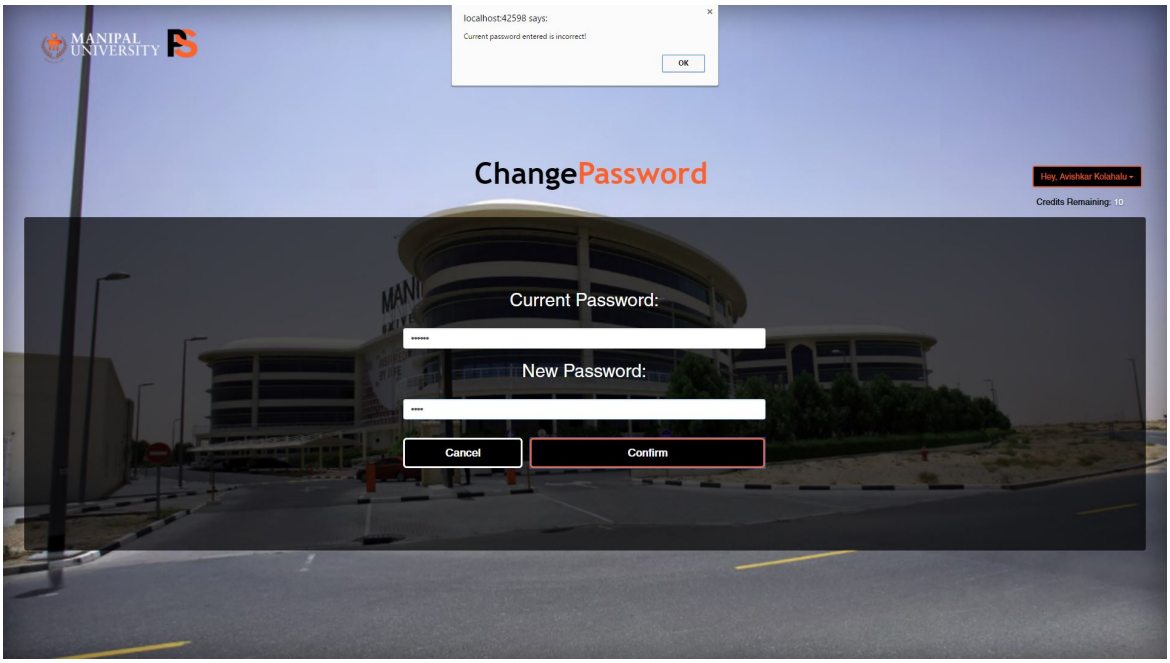

Figure 4.15: PrintStation – Student – Password Validation Page

An alert is shown when the new password does not meet the validations set.

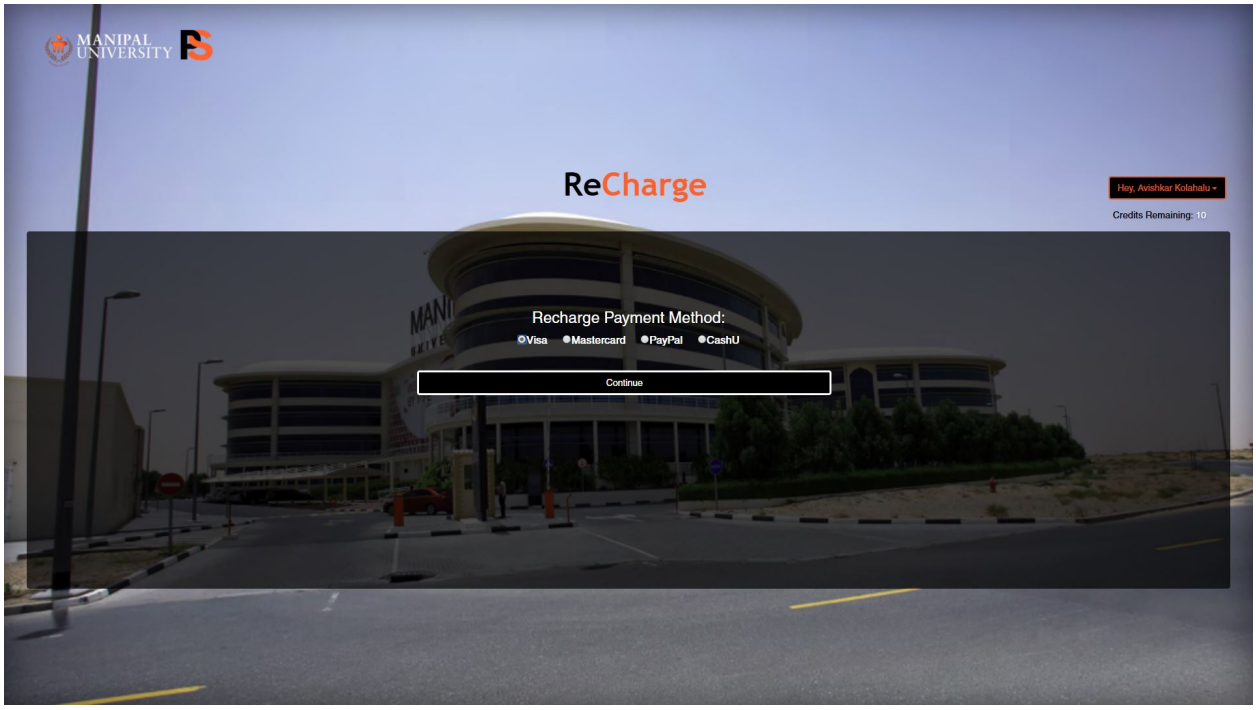

Figure 4.16: PrintStation – Student – Recharge Page

The student can recharge their account on this page when they have insufficient credit.

4.3.3. Faculty Dashboard

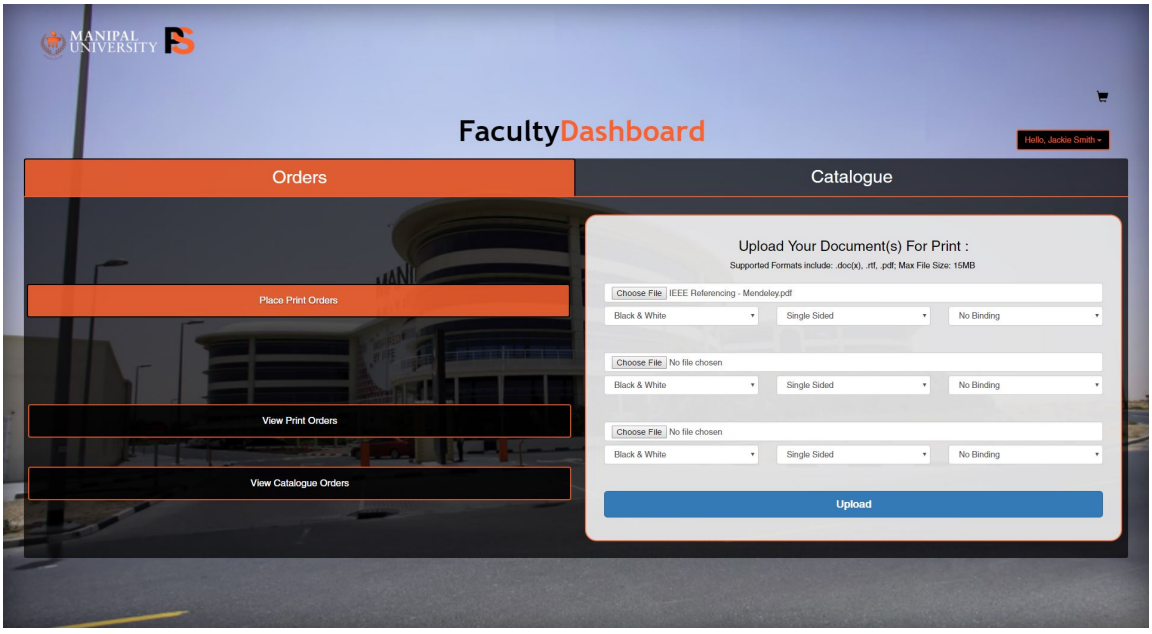

Figure 4.17: PrintStation – Faculty – Dashboard

This is the first page which opens when faculty logs in.

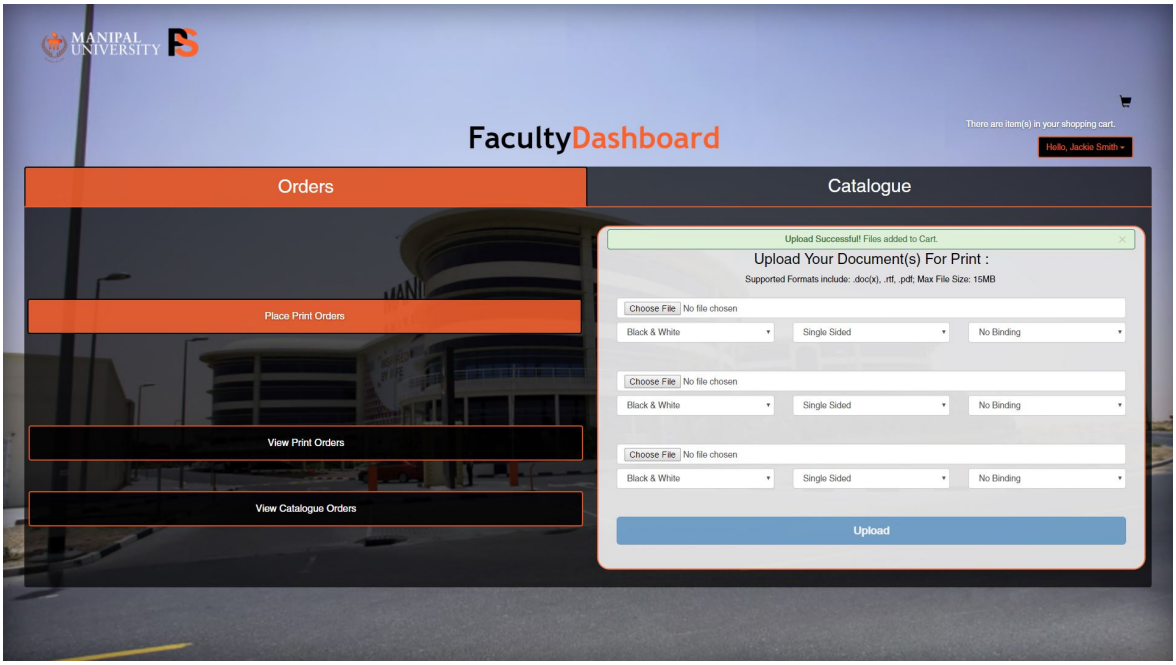

Figure 4.18: PrintStation – Faculty – Place Print Orders Page

The faculty can place upload the files they wish to print on this page. They can choose the kind of print they want i.e. if they want it in colour, black and white, single sided, double sided or if they want it to be bound.

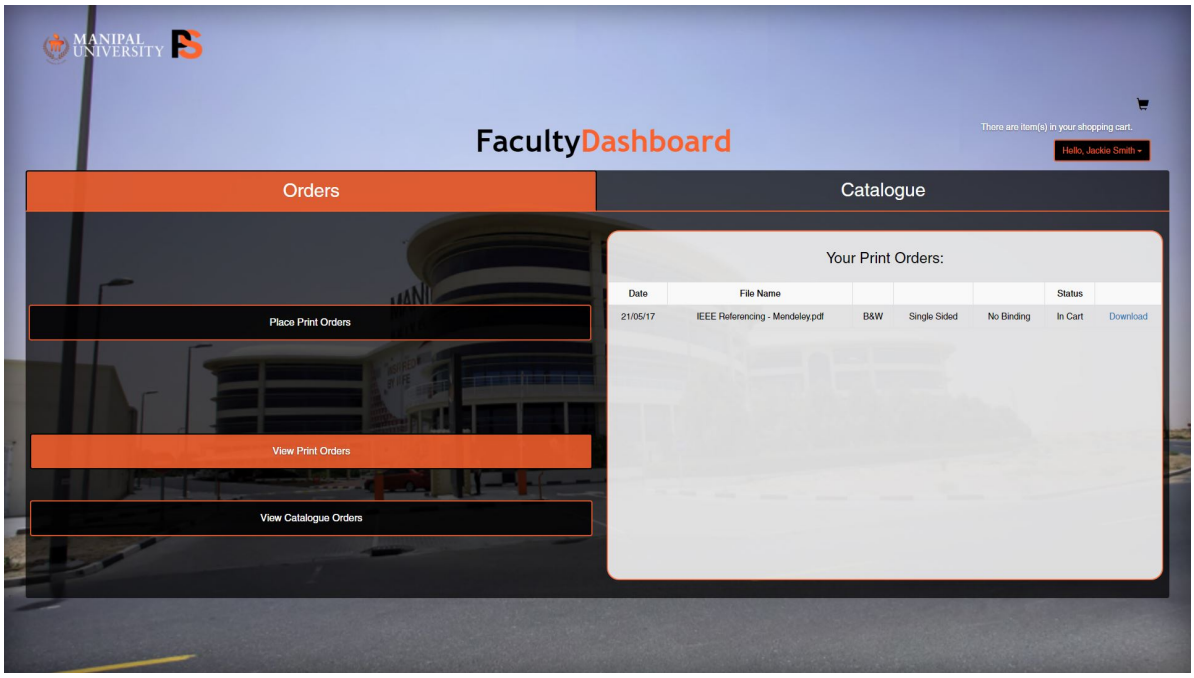

Figure 4.19: PrintStation – Faculty – View Print Order Page

This page shows that the print orders the faculty has given so far. It shows the status for every order made.

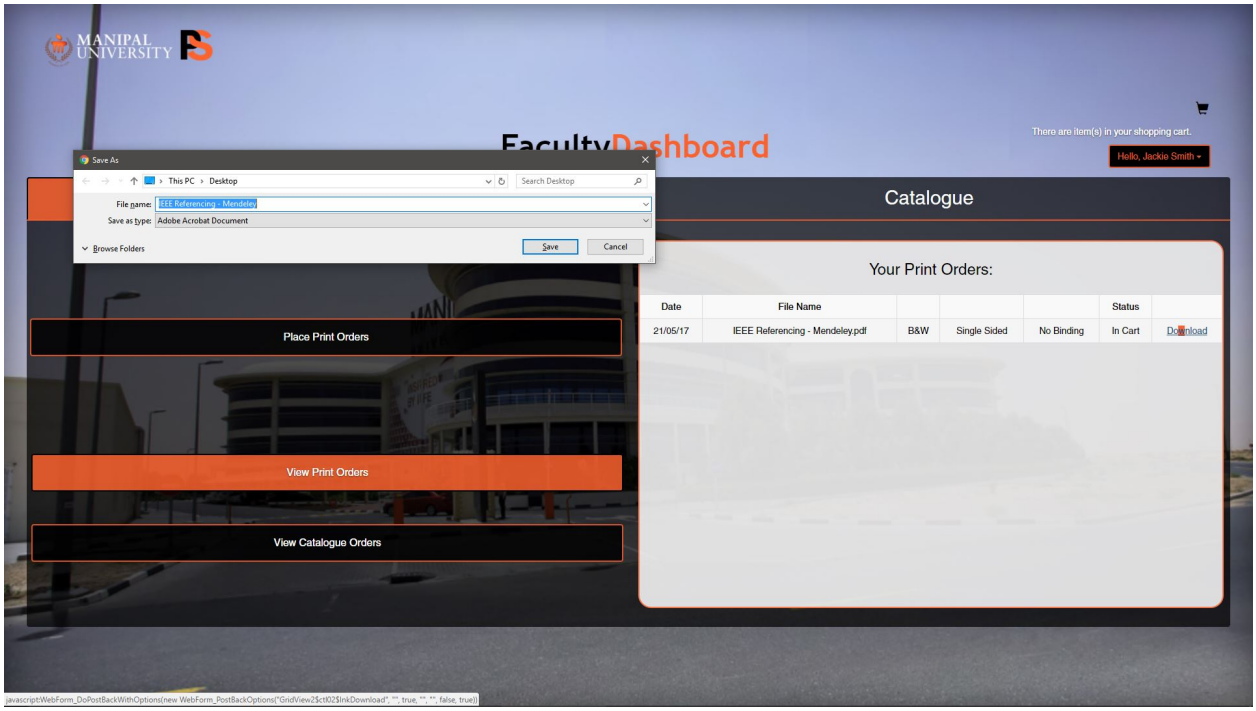

Figure 4.20: PrintStation – Faculty – Download File

The print orders that have been made by the users can be downloaded by the user anytime.

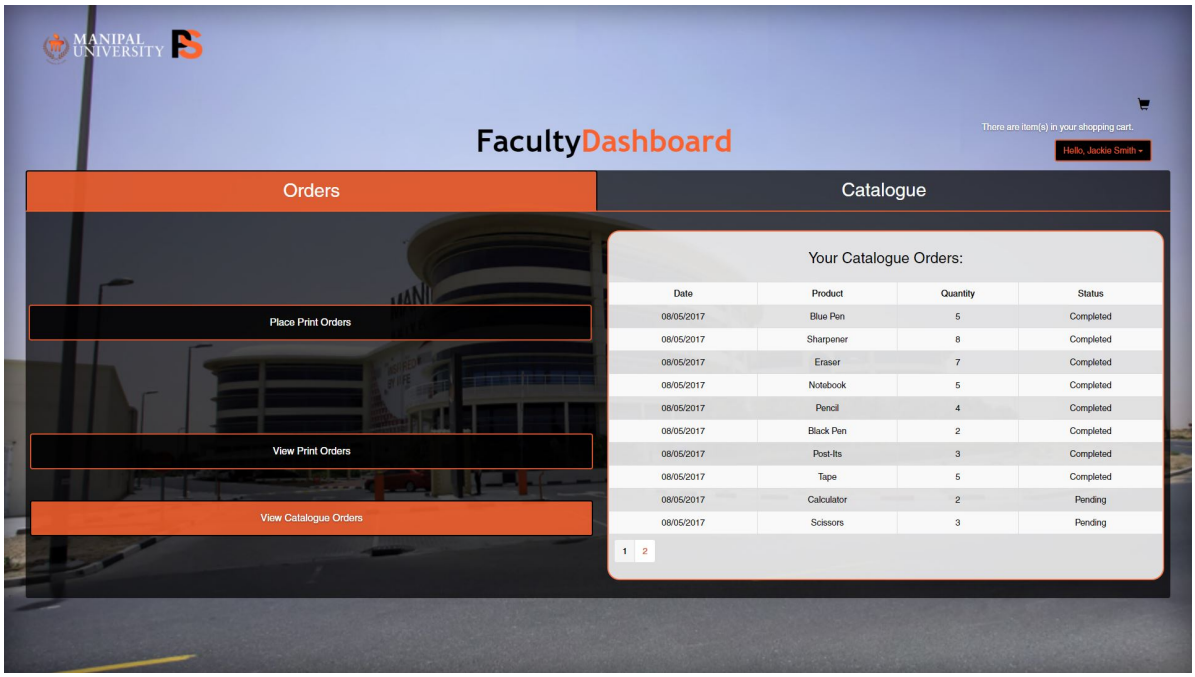

Figure 4.21: PrintStation – Faculty – View Catalogue Orders Page

This page shows that the shop orders the faculty has given so far. It shows the status for every order made.

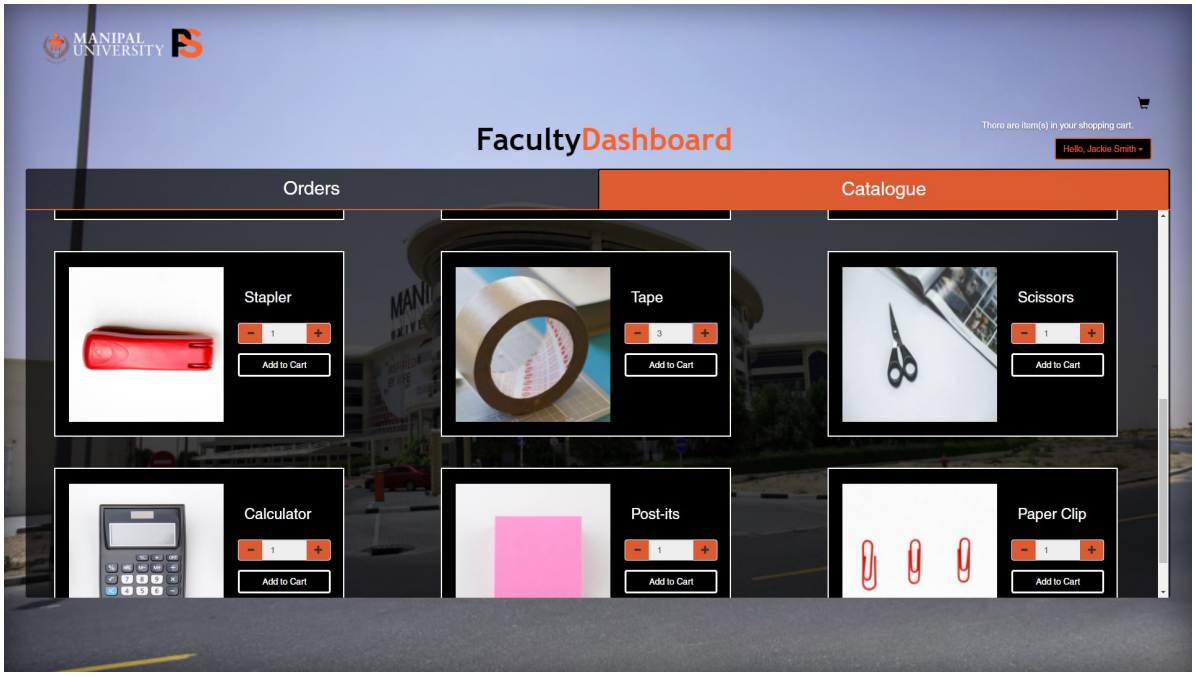

Figure 4.22: PrintStation – Faculty – Catalogue Page

This page shows the shop catalogue for the faculty to buy what they require.

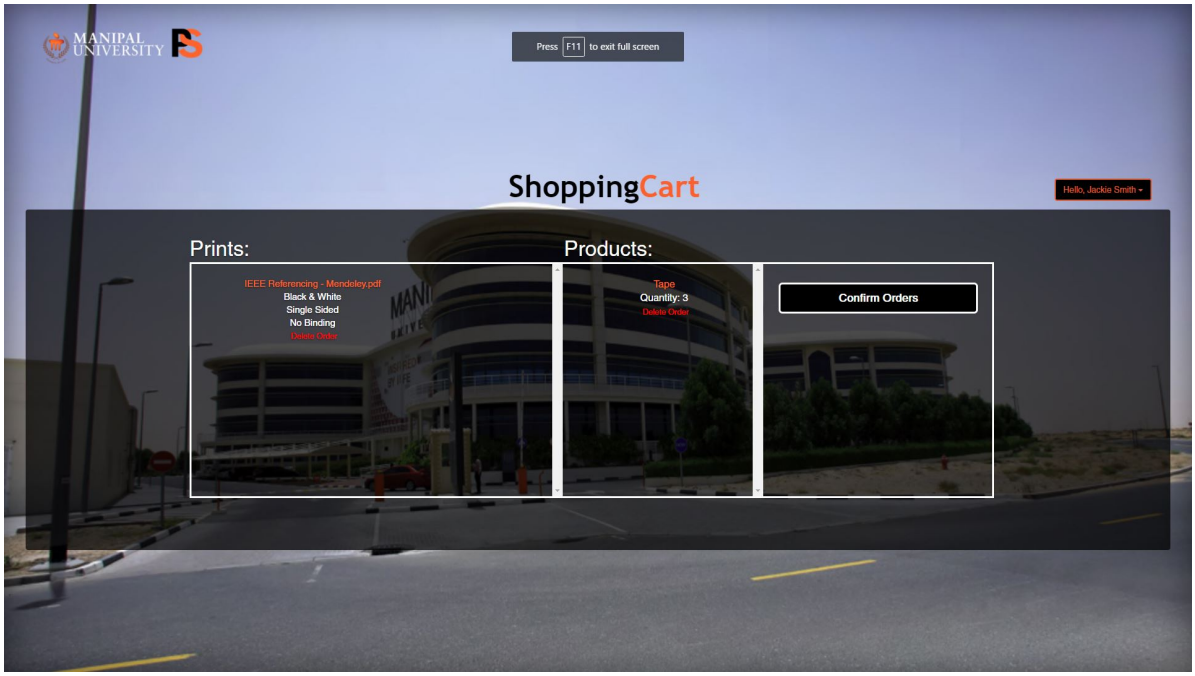

Figure 4.23: PrintStation – Faculty – Shopping Cart Page

This page shows all the items added to the shopping cart. It shows all the items with the quantity before order confirmation. It does not show the prices for the items since the faculty do not have to pay for their purchases.

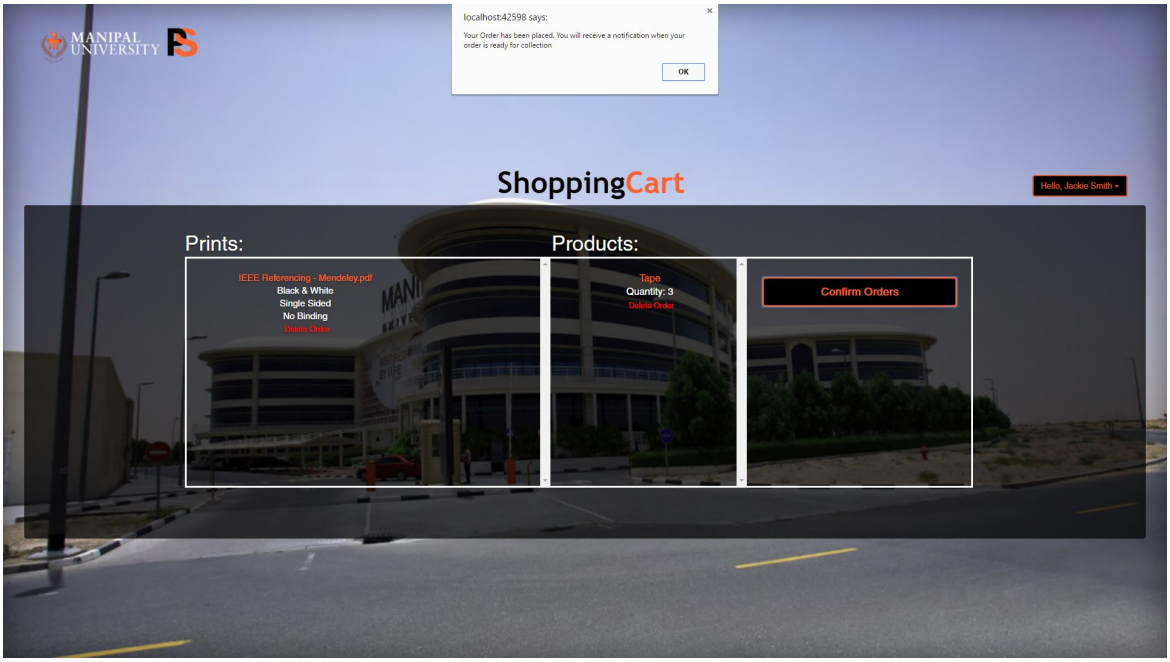

Figure 4.24: PrintStation – Faculty – Cart Checkout

Once the order is made an alert is shown stating the order has been made successfully.

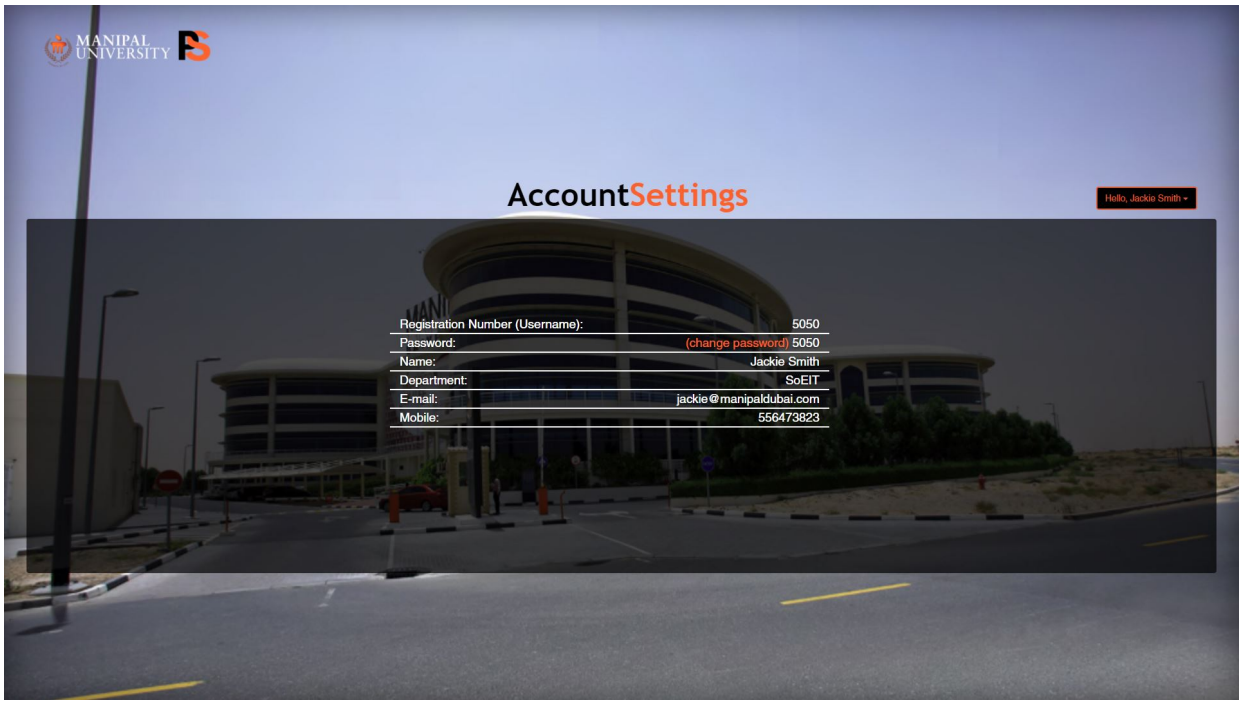

Figure 4.25: PrintStation – Faculty – Account Setting Page

This page shows the details of the faculty. The user can change password on this page.

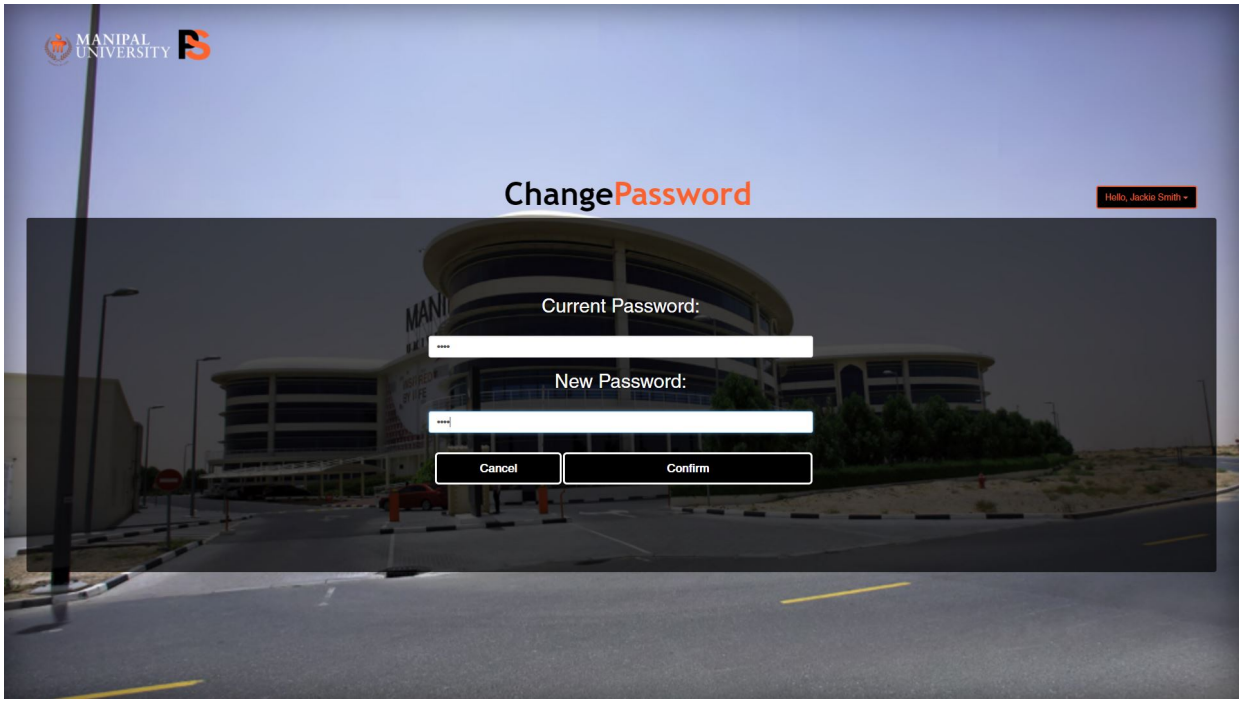

Figure 4.26: PrintStation – Faculty – Change Password Page

This page allows the user to create a new password for their account.

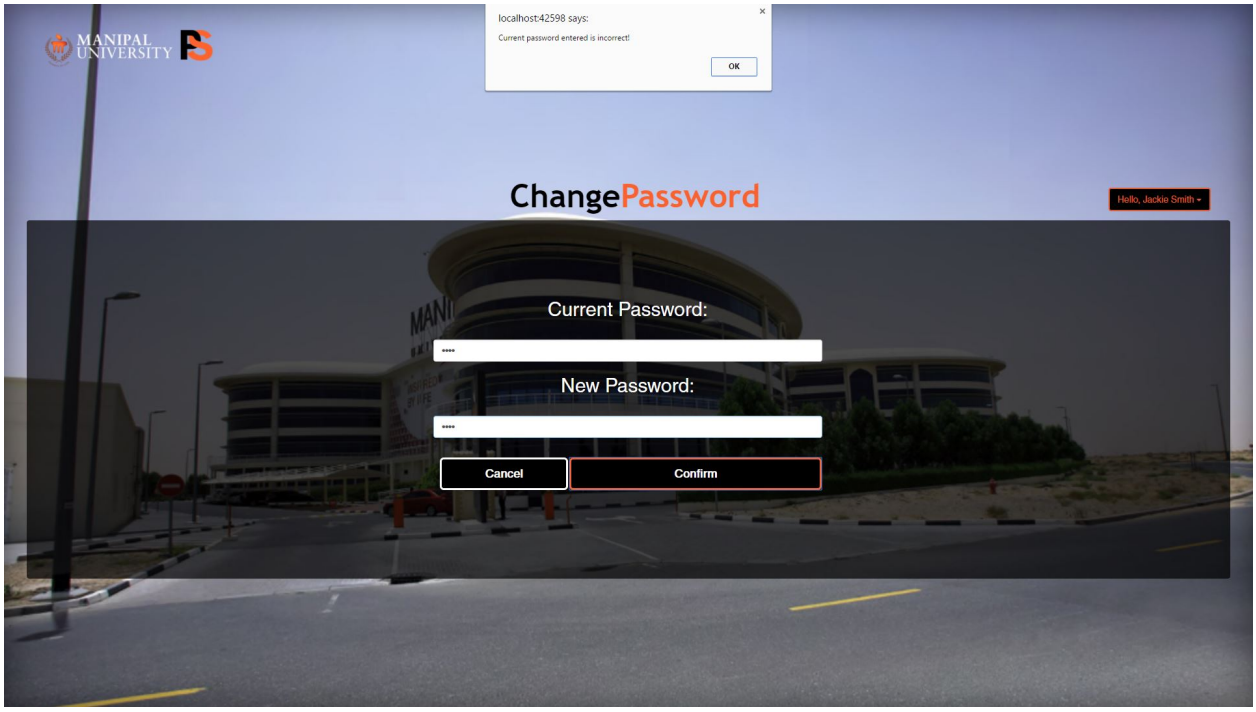

Figure 4.27: PrintStation – Faculty – Password Validation

An alert is shown when the new password does not meet the validations set.

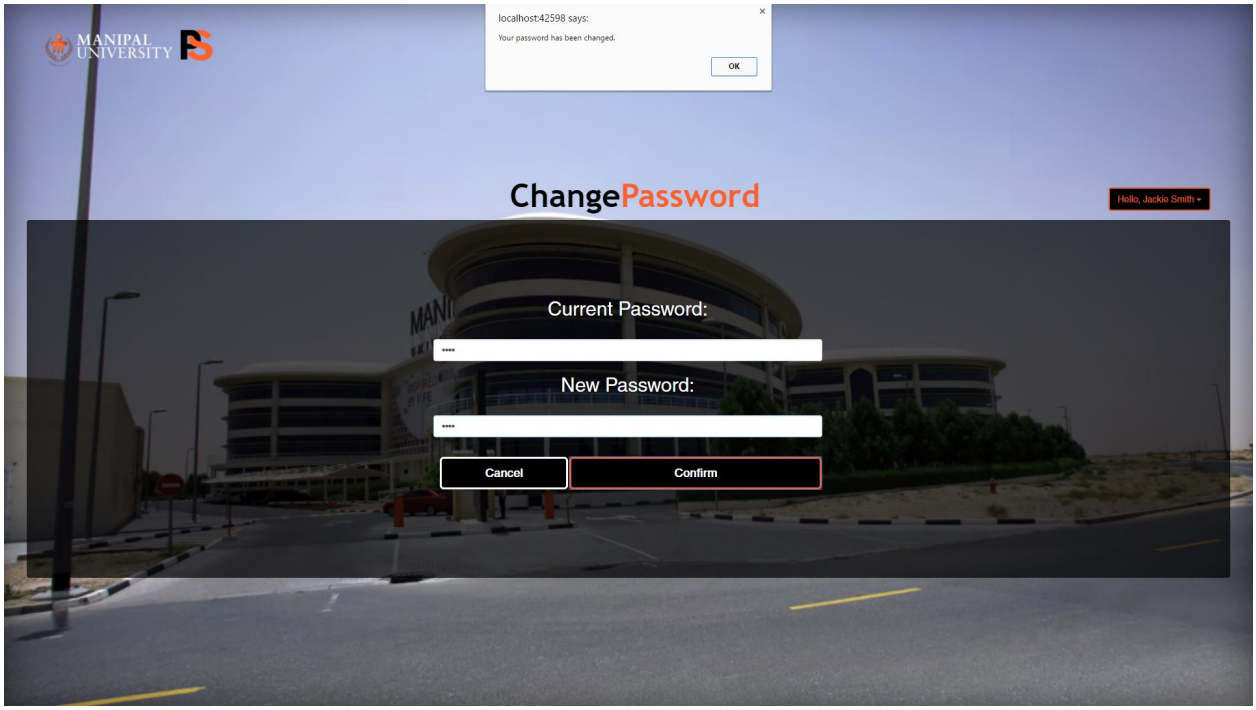

Figure 4.28: PrintStation – Faculty – Password Changed

This page notifies the faculty that the password has been changed.

4.3.4. Admin Dashboard

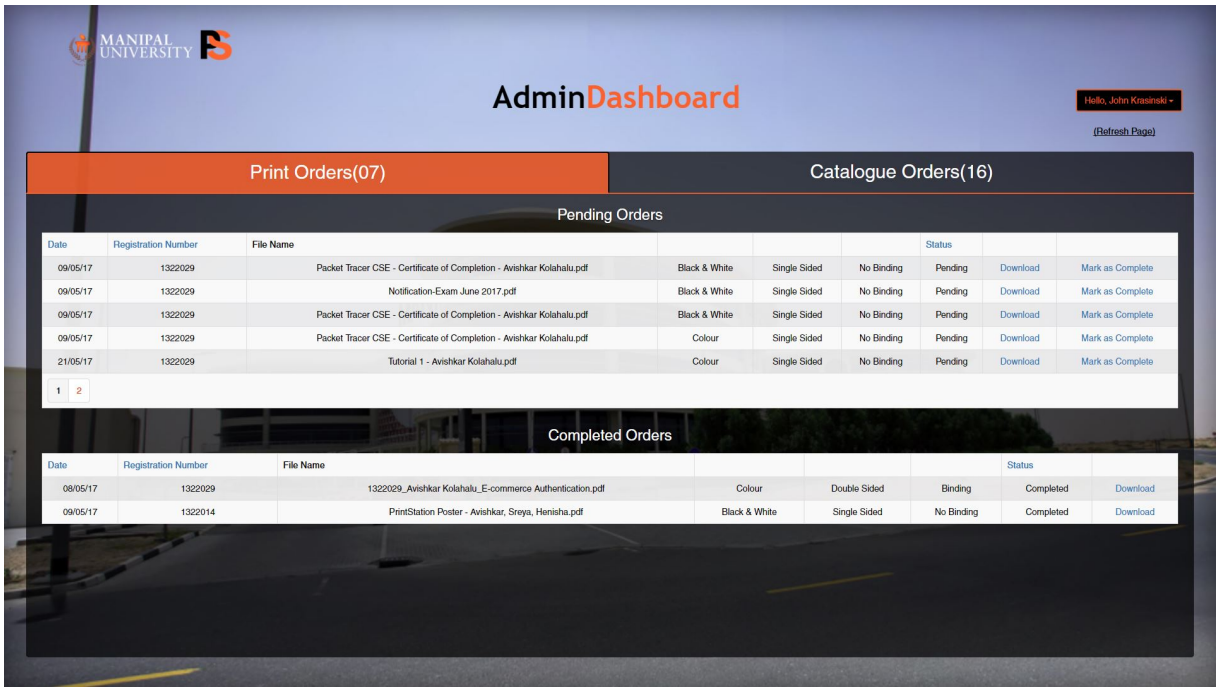

Figure 4.29: PrintStation – Admin – Dashboard

This is the first page that opens when the user logs in. The page has two parts- Print Orders and Catalogue Orders.

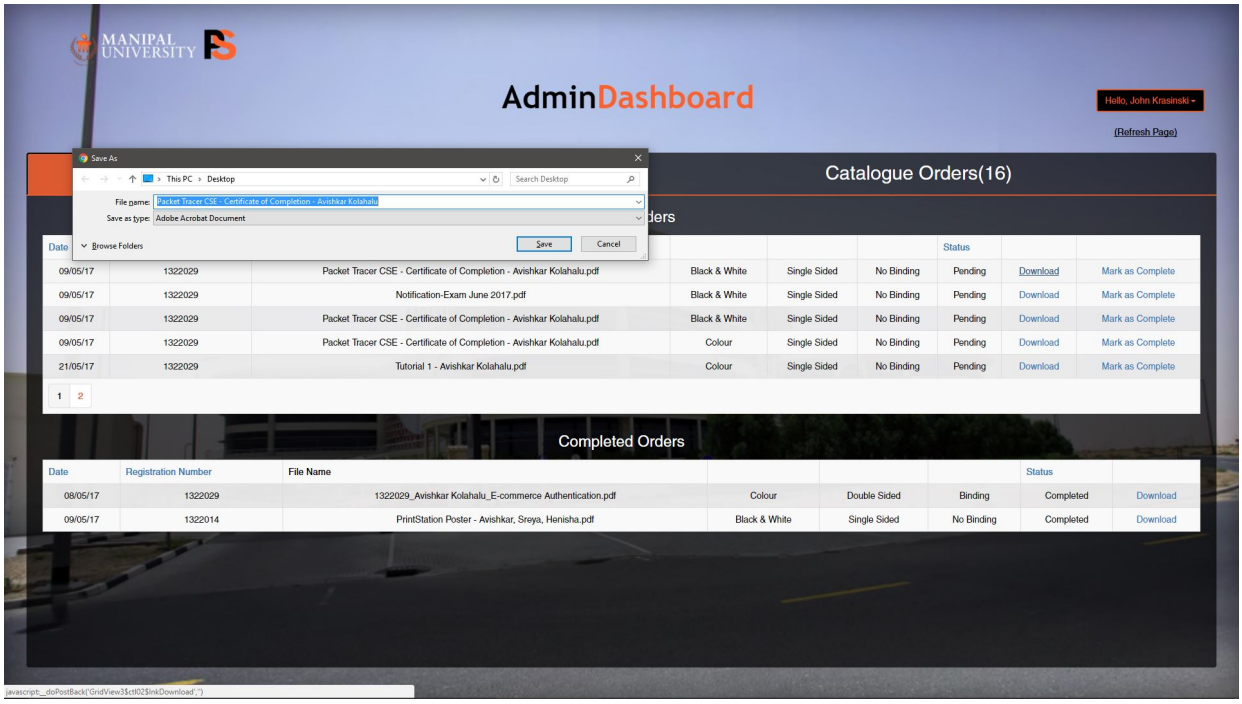

Figure 4.30: PrintStation – Admin – Download for print

The admin can download the orders made by the users for printing.

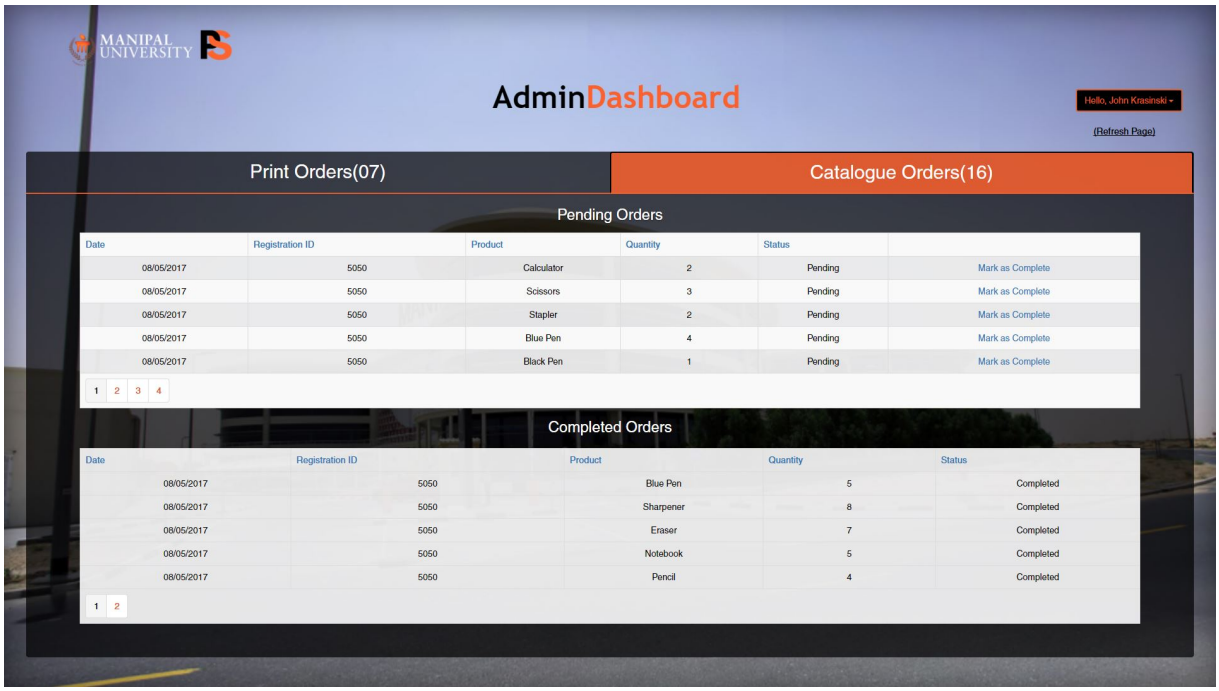

Figure 4.31: PrintStation – Admin – Catalogue Orders

This page shows all the catalogue orders made by the users. It shows them as two parts- pending orders and completed orders. The admin can mark an order as complete when he gets the items ready for the user to come pick up.

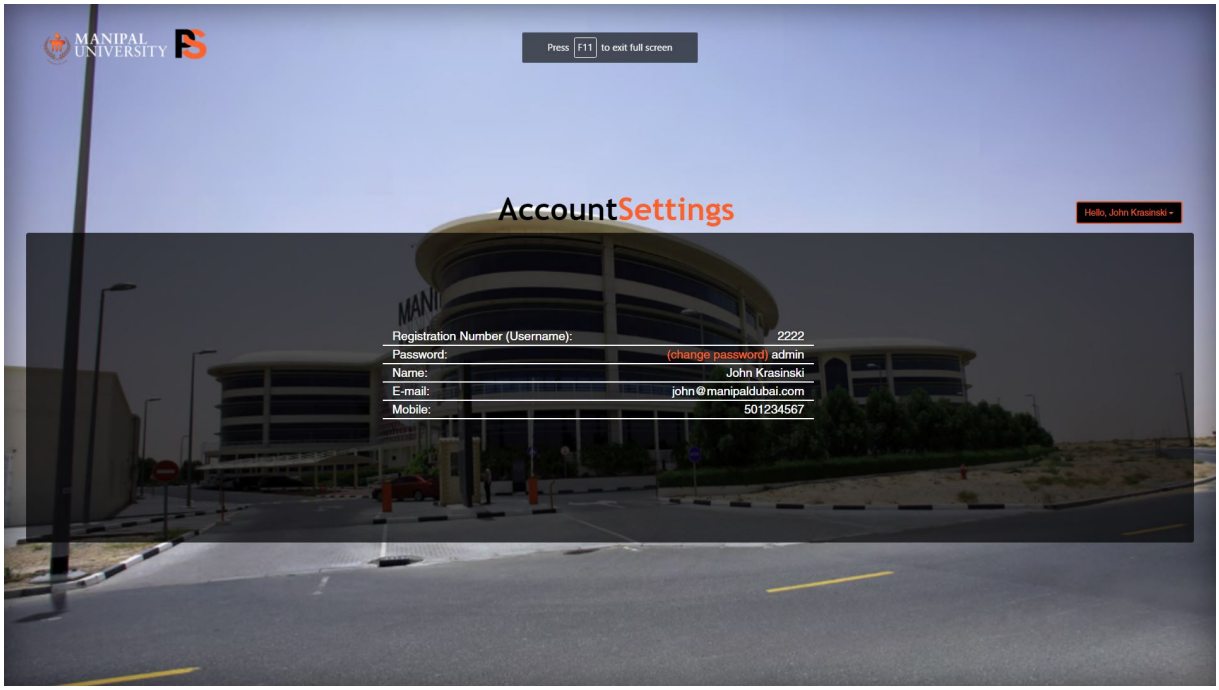

Figure 4.32: PrintStation – Admin – Account Settings

It shows all the details of the administrator.

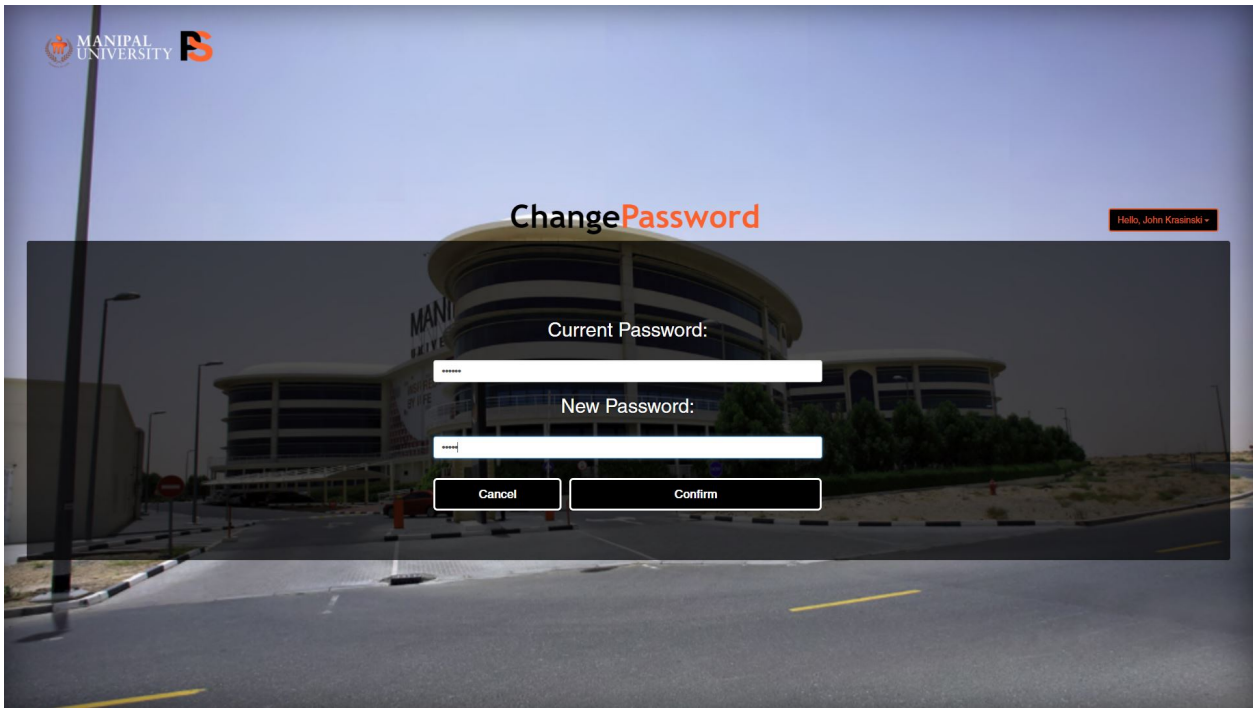

Figure 4.33: PrintStation – Admin – Change Password Page

This page allows the user to create a new password for their account.

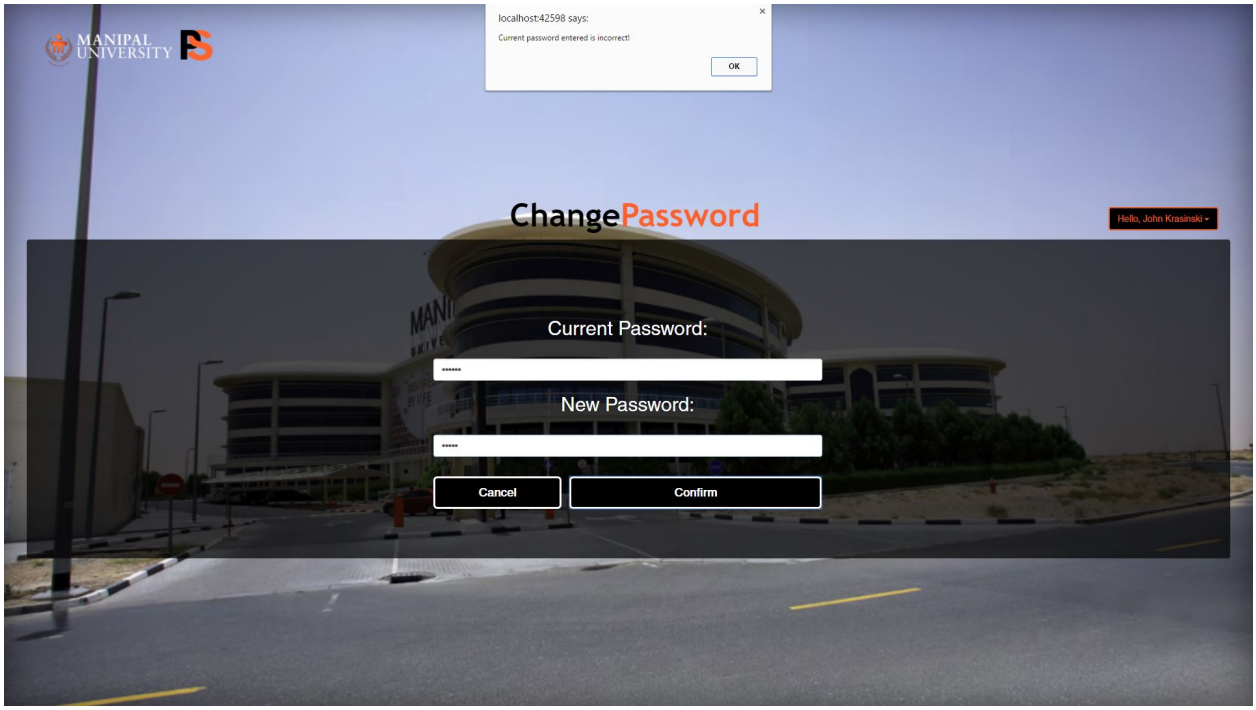

Figure 4.34: PrintStation – Admin – Password Validation

An alert is shown when the new password does not meet the validations set.

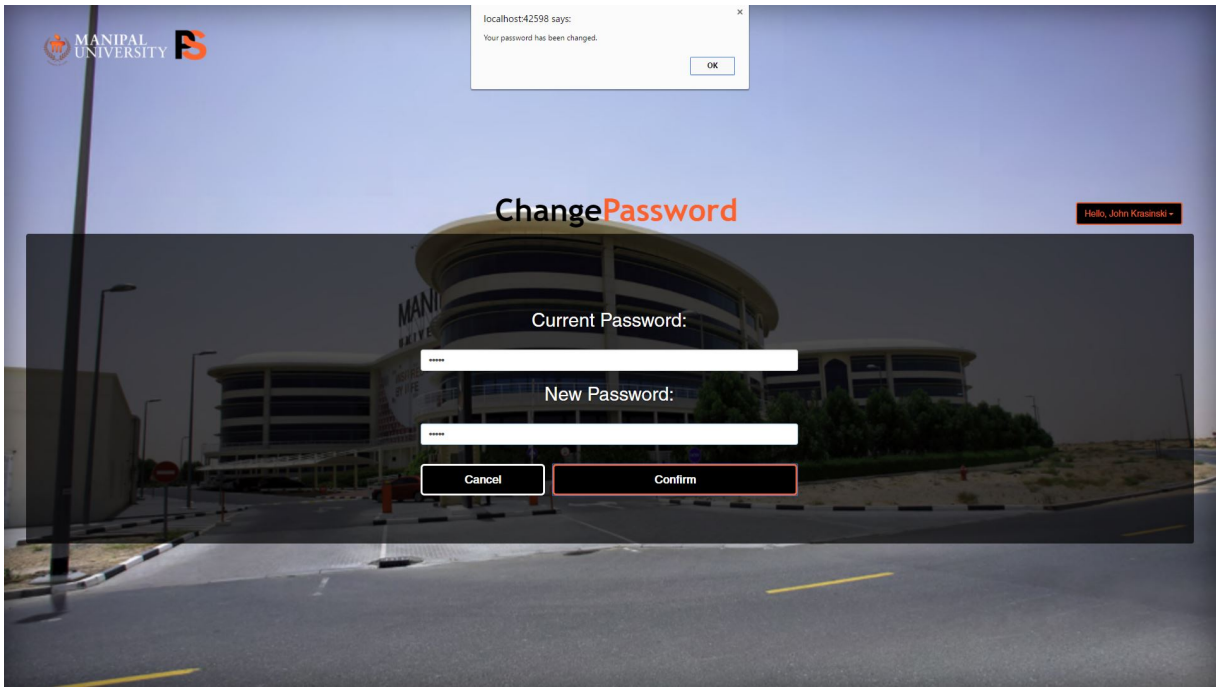

Figure 4.35: PrintStation – Admin – Password Changed

This page notifies the faculty that the password has been changed.

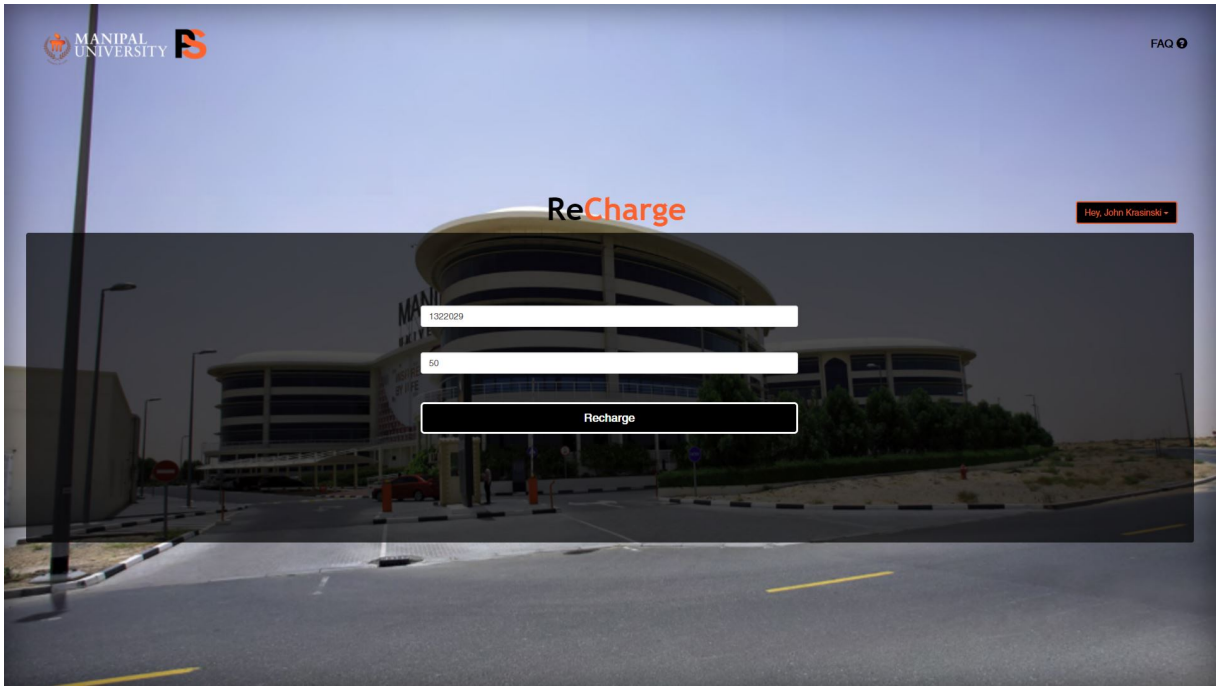

Figure 4.36: PrintStation – Admin – Recharge Page

This page allows the admin to recharge the student accounts when the student do not have sufficient credits and they wish recharge. When the student pays the admin with cash, the equivalent number of credits would be added to their account.

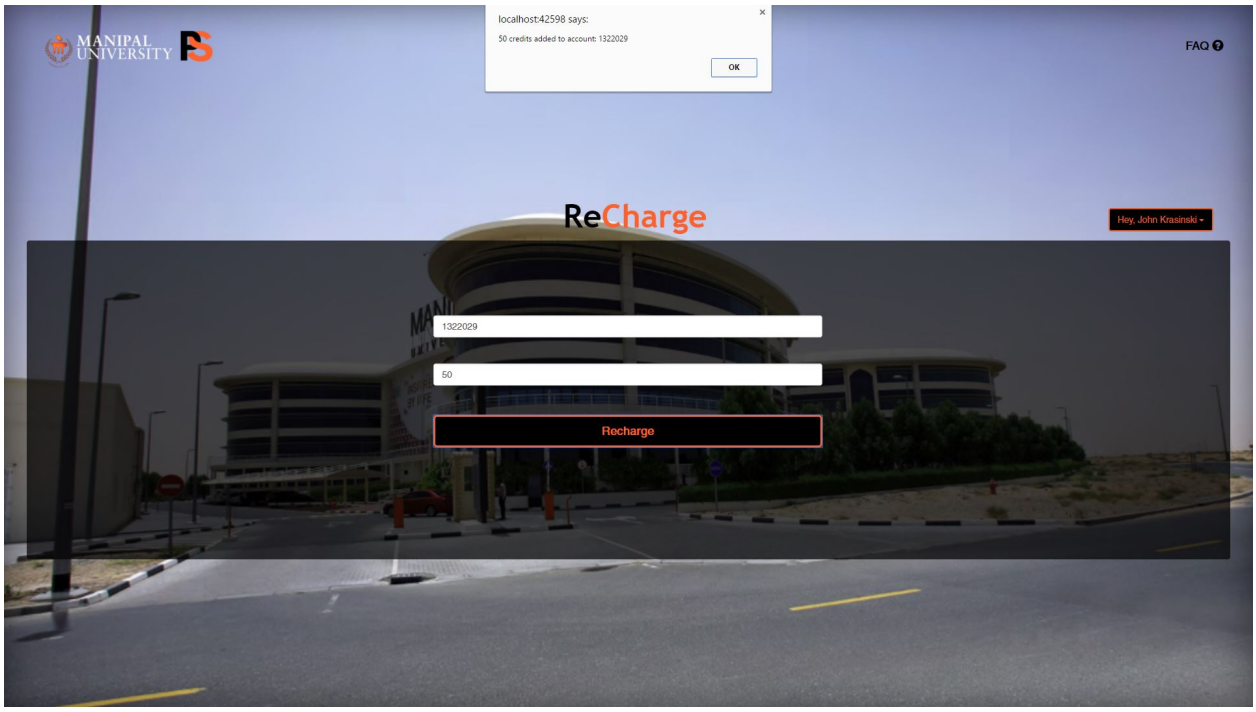

Figure 4.37: PrintStation – Admin – Account Recharged

An alert is showed when the recharging of the student account is done successfully.

4.4. Screenshots for Android Application

4.4.1 Home Page

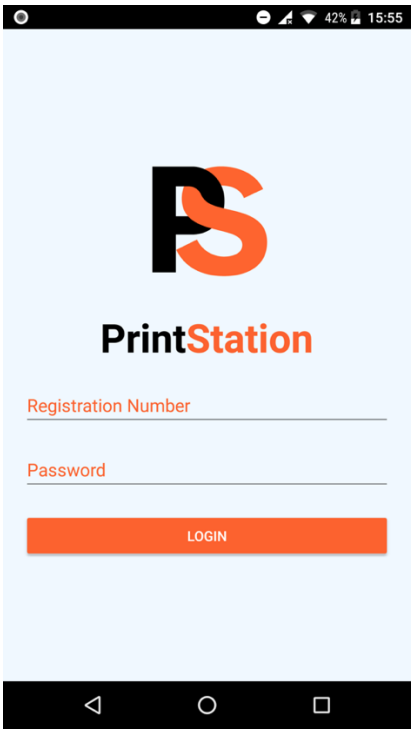

Figure 4.38: PrintStation Mobile – Login

4.4.2. Student Dashboard

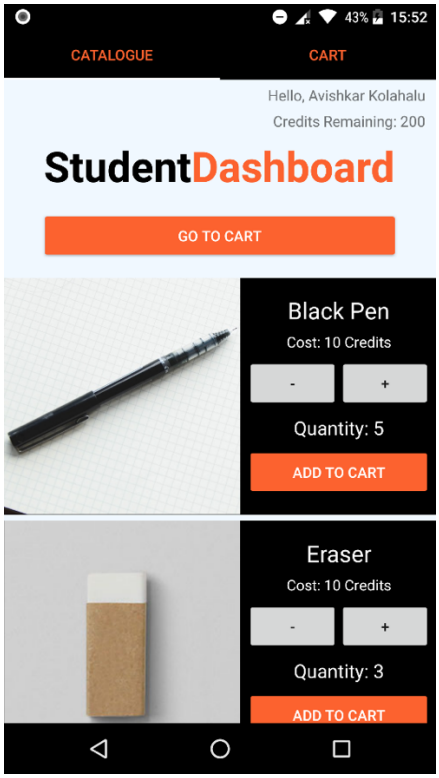

Figure 4.39: PrintStation Mobile – Student Dashboard

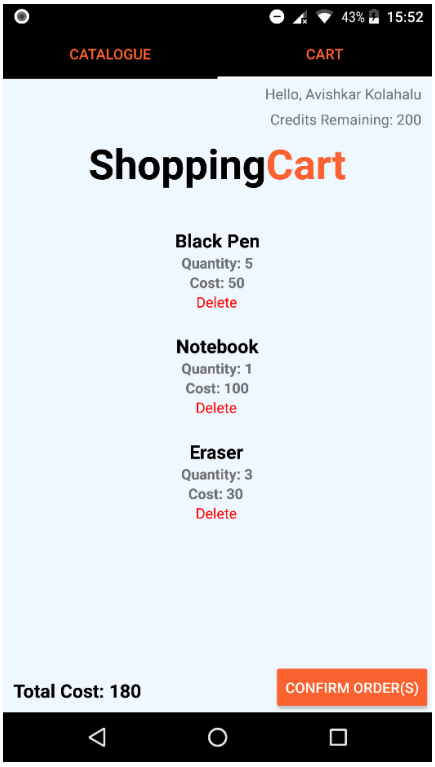

Figure 4.40: PrintStation Mobile – Student Cart

4.4.3. Faculty Dashboard

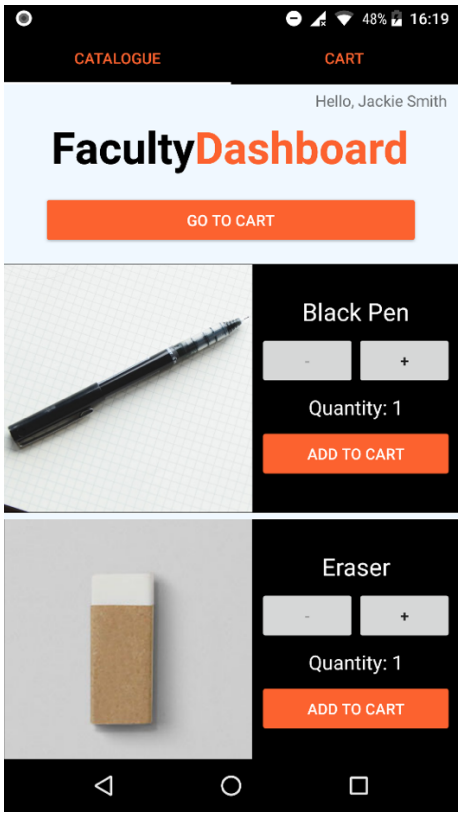

Figure 4.41: PrintStation Mobile – Faculty Dashboard

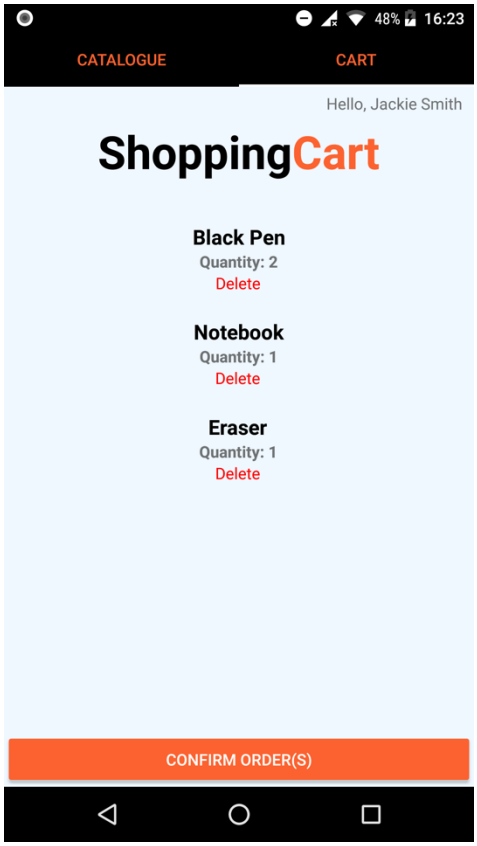

Figure 4.42: PrintStation Mobile – Faculty Cart

5. Future Scope

Being university students not every student has access to a credit card. There is the facility to recharge credits by paying the admin at the photocopy centre. But in future the university could have re-charging stations around the campus so that students can do it by themselves instead of waiting for the admin to do it. Since the main aim of PrintStation is to avoid queues.

Currently the PrintStation consists only has a web application and Android application. We intend to create an iOS application as well to reach all those users with Apple devices.

The university has many systems for different necessities like for the library, the accounts, the student portal. PrintStation could be integrated with these systems. The credit based payments which is introduced in PrintStation could be implemented with the other systems as well. This would help the users have one place to make all their payments.
For example, if the student has any dues in the library for not returning the book issued on time, it can be paid using the credits bought on PrintStation.

6. Conclusion

PrintStation benefits users in terms of time management. The users need not wait in queues. This application will make printing jobs easier and provide an easy access to buy stationery available online. Instead of the tedious process of collecting USBs or Hard drives, and sorting through e-mail to obtain the files for print, the admin would easily be able to view all the orders through the system on the web portal, and files that require printing.

References:

[1] Saleh, K. (2009). Software Engineering. 1st ed. J. Ross Publishing.

[2] Techterms.com. (2017). ASP.NET Definition. [online] Available at: https://techterms.com/definition/aspnet [Accessed 21 May 2017].

[3] Hermes, D. (2015). Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals. 1st ed. Apress.

[4] W3schools.com. (n.d.). SQL Introduction. [online] Available at: https://www.w3schools.com/sql/sql_intro.asp [Accessed 20 May 2017].

[5] Rumbaugh, J., Jacobson, I. and Booch, G. (2005). Advanced Praise for The Unified Modeling Language Reference Manual. 2nd ed. Addison-Wesley.