Implementation Document

for

Unified Portal for Hall Automation

Version 1.0

Prepared by

Group #: 1 **Group Name: The Tech Titans**

Soham Amit Bharambe	**210264**	**sohamb21@iitk.ac.in**
Divyansh	**210355**	**divyansh21@iitk.ac.in**
Divyansh Chhabria	**210356**	**divyanshc21@iitk.ac.in**
Jhalak Sharma	**210474**	**jhalak21@iitk.ac.in**
Kriti	**210534**	**kriti21@iitk.ac.in**
Kumar Harsh Mohan	**210543**	**harshmohan21@iitk.ac.in**
Labajyoti Das	**210552**	**labajyoti21@iitk.ac.in**
Pranjal Singh	**210744**	**psingh21@iitk.ac.in**
Rajeev Kumar	**210815**	**rajeevks21@iitk.ac.in**
Sandeep Nitharwal	**210921**	**nsandeep21@iitk.ac.in**

Course: **CS253**

Mentor TA: **Mr. Ashitosh Vankatrao More**

Date: **20th March, 2023**

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Soham Amit Bharambe Divyansh Divyansh Chhabria Jhalak Sharma Kriti Kumar Harsh Mohan Labajyoti Das Pranjal Singh Rajeev Kumar Sandeep Nitharwal	First draft	20/03/2023

1 Implementation Details

Programming Language, Framework and Libraries for:

Backend

Unified Portal for Hall Automation (UPHA) has been mainly written in **Python**. We have used **Django**, a Python-based free and open-source framework for web development. It follows the *model-view-controller* architectural pattern. We chose Django because it is **robust yet simple**, reliable, and scalable. There are numerous advantages of Django, which are briefly stated below:

- Django's syntax is simple and easy to learn, making it an attractive option for web development. Its code is written in Python.
- It has its own built-in server.
- Django has built-in features to prevent SQL injection, cross-site scripting (XSS) attacks and robust user authentication system, ensuring that web apps built are secure and reliable.
- Django has an object-oriented design, which promotes code reuse and modularity.
- Django is scalable and can handle high web traffic efficiently. This allows Django to serve many users simultaneously without experiencing performance issues.
- Django comes with a built-in admin interface for developers, making it easy to manage data models and view data from the app.
- Django supports rapid development by providing tools for scaffolding out the basic structure of a web application.
- It comes with a built-in database (SQLite), which is easy to set up and use.
- It encourages modularity of code.
- It has excellent community support, with many developers contributing to the framework and sharing their expertise through online forums and documentation.

We have also used many Python and Django libraries, a detailed list of which is given in the github repository.

Frontend

- We have used **HTML** (HyperText Markup Language) as a standard markup language for web pages to be displayed in the web browser. It provides a basic structure for web content, including text, images and other media.
- We have used **CSS** (Cascading Style Sheets) for styling the web pages.
- It allows for the separation of content and presentation, making it easier to maintain and update the web pages.

Database

- We have used the **SQLite Database Engine**. It comes built-in with Django and is small, fast, and highly reliable.
- It is a widely-used and one of the most popular relational database engines in the world.
- Due to its file-based design, it eliminates the need for a separate database server, making it a simple and efficient choice.

2 Codebase

Github Repository: http://github.com/divc13/Unified-Portal-For-Hall-Automation

Code Structure

The project has been divided into following small apps:

1. **Mess**: It contains implementation of functionalities related to mess, i.e., student can view regular menu, book extras, view order history, and apply for rebate, and owner can modify regular menu and extras, react on rebate requests, view students' bill, view extras orders, view feedback on regular menu, and modify BDMR.
2. **Canteen:** It contains implementation of functionalities related to canteen, i.e., students can place order, view pending order, and view order history, and owner can react to orders, modify menu, and view students' bill.
3. **Booking**: It contains implementation of functionalities related to booking, i.e., guest room booking, sports equipment issue, and sports court booking.
4. **Cleaning**: It contains implementation of functionalities related to cleaning, i.e., lodge request and view past requests.
5. **Login**: It contains implementation of functionalities related to login, i.e., login old user, new user registration, and reset password.
6. **My_Account**: It contains implementation of accounts management of the students, i.e., mess bill, canteen bill, and electricity bill.
7. **Home**: It contains implementation of the homepage.

Three more folders and two more files are also present in the repository:

1. **templates**: It contains HTML files of base template.
2. **UPHA**: It contains files which are necessary to run the web app.
3. **manage.py**: A command-line utility that lets us interact with this Django project in various ways such as run server, create new app, etc.

Each app mentioned above contains following folders and files:

1. **migrations**: It propagates changes we make to our models (adding a field, deleting a model, etc.) into our database schema.
2. **templates**: It contains HTML files of web pages of the respective app.
3. **static (present only in Home and Login)**: It contains CSS files of web pages and also images related to them of the respective app.
4. **__init.py__**: It is an empty file that tells Python that this directory should be considered a Python package.

5. **admin.py**: It contains the administration interface for our app. The admin interface is a web-based interface that allows authorized users to manage the data in your app, including creating, editing, and deleting records.
6. **apps.py**: It contains the configuration for a specific Django app. It is used to customize various aspects of the app's behavior.
7. **models.py**: It contains the structure of our database tables using Python classes. Each class represents a database table, and the attributes of the class define the fields of the table.
8. **tests.py**: It is a module where we write test cases for our application. It should contain one or more classes that inherit from Django's TestCase class. Each class represents a group of related tests, and each method within the class represents a specific test.
9. **urls.py**: It contains the URL patterns for our application. Each URL pattern maps a URL to a specific view, which generates the HTTP response for that URL.
10. **views.py**: It contains the logic for handling requests and generating responses. Each view function takes a request object as its first argument and returns an HTTP response object.

Directories

```
|── .gitignore
|── Booking
|    |── admin.py
|    |── apps.py
|    |── migrations
|        |── _init_.py
|    |── models.py
|    |── templates
|        |── booking_error.html
|        |── Booking_manager.html
|        |── courts.html
|        |── guestroom.html
|        |── secy_add_equipments.html
|        |── secy_validate_request.html
|        |── secy_validate_return.html
|        |── sports_equipments.html
|    |── tests.py
|    |── urls.py
|    |── views.py
|    |── _init_.py
|── Canteen
|    |── admin.py
|    |── apps.py
|    |── migrations
```

```
|         |── _init_.py
|     |── models.py
|     |── templates
|         |── Owner_Modify_Menu.html
|         |── Owner_New_Order.html
|         |── Owner_Pending_Order.html
|         |── Owner_Students_Bill.html
|         |── Student_Cart.html
|         |── Student_Orders_History.html
|         |── Student_Pending_Order.html
|         |── Student_Place_Order.html
|     |── tests.py
|     |── urls.py
|     |── views.py
|     |── _init_.py
|── Cleaning
|     |── admin.py
|     |── apps.py
|     |── migrations
|         |── _init_.py
|     |── models.py
|     |── templates
|         |── Cleaning_hall.html
|         |── Lodge_Request.html
|         |── Past_Request.html
|         |── Pending_Request.html
|     |── tests.py
|     |── urls.py
|     |── views.py
|     |── _init_.py
|── Home
|     |── admin.py
|     |── apps.py
|     |── migrations
|         |── _init_.py
|     |── models.py
|     |── static
|         |── hall_image.jpg
|     |── templates
|         |── Home.html
```

```
|     |── tests.py
|     |── urls.py
|     |── views.py
|     |── _init_.py
|── Login
|     |── admin.py
|     |── apps.py
|     |── migrations
|         |── _init_.py
|     |── models.py
|     |── static
|         |── Login
|             |── upha.png
|             |── vector.png
|     |── templates
|         |── Login.html
|         |── OTP.html
|         |── Reset_Password.html
|         |── Set_Password.html
|         |── SignUp.html
|     |── tests.py
|     |── urls.py
|     |── views.py
|     |── _init_.py
|── manage.py
|── Mess
|     |── admin.py
|     |── apps.py
|     |── migrations
|         |── _init_.py
|     |── models.py
|     |── templates
|         |── Manager_Extra_Items.html
|         |── Manager_Modify_BDMR.html
|         |── Manager_Modify_Menu.html
|         |── Manager_Rebate_Requests.html
|         |── Manager_Students_Pending_Bills.html
|         |── Manager_View_Feedback.html
|         |── Manager_View_Orders.html
|         |── Student_Apply_For_Rebate.html
```

```
|         |── Student_Booked_Extras.html
|         |── Student_Book_Extras.html
|         |── Student_Order_History.html
|         |── Student_Regular_Menu.html
|     |── tests.py
|     |── urls.py
|     |── views.py
|     |── _init_.py
|── My_Account
|     |── admin.py
|     |── apps.py
|     |── migrations
|         |── _init_.py
|     |── models.py
|     |── templates
|         |── Canteen.html
|         |── Mess.html
|     |── tests.py
|     |── urls.py
|     |── views.py
|     |── _init_.py
|── templates
|     |── error.html
|     |── staff_base.html
|     |── static
|         |── booking_1_1.png
|         |── football.png
|         |── home_1.png
|         |── hostel.png
|         |── household_1.png
|         |── logo_black_1.png
|         |── main_display.png
|         |── power_1.png
|         |── rectangle_5.png
|         |── staff_base.css
|         |── student_base.css
|         |── styles.css
|         |── wallet_1.png
|     |── student_base.html
|── UPHA
```

```
|     |── asgi.py
|     |── settings.py
|     |── urls.py
|     |── wsgi.py
|     |── _init_.py
```

API Endpoints

1. Login

a. Login

```
URL: /

Method: POST

Data: {
    username,
    password
}

Status: {
    If successful: {
        200_OK
        302_Found (Redirect to home page)
    }
    Else: {
        400_Bad Request (Already logged in)
        401_Unauthorized (Wrong password/ Not registered)
    }
}
```

b. Set password

```
URL: /Set_Password

Method: POST

Data: {
    username,
    password,
    OTP
}

Status: {
    If successful: {
        200_OK
        302_Found (Redirect to login page)
    }
    Else: 400_Bad Request (Password doesn't fulfill criteria)
}
```

c. Reset password

```
URL: /Reset_Password

Method: POST

Data: {
    username,
    password,
    OTP
}

Status: {
    If successful: {
        200_OK
        302_Found (Redirect to login page)
    }
    Else: 400_Bad Request (OTP is not correct)
}
```

d. Sign Up

```
URL: /SignUp

Method: POST

Data: {
    name,
    username,
    designation,
}

Status: {
    If successful: {
        201_Created
        302_Found (Redirect to login page)
    }
    Else: 400_Bad Request (Already registered/ Some required data is
missing)
}
```

e. OTP

```
URL: /OTP

Method: POST

Data: {
    username
}

Status: {
    If successful: {
        200_OK
        302_Found (Redirect to set password page)
    }
    Else: 400_Bad Request (username is missing)
}
```

f. Logout

```
URL: /Logout

Method: POST

Data: {
    username
}

Status: {
    If successful: {
        200_OK
        302_Found (Redirect to login page)
    }
    Else: 401_Unauthorized (Not logged in)
}
```

2. Canteen

a. Student add item to cart

```
URL: /Canteen/Student_Place_Order

Method: POST

Data: {
    User_Name,
    Name,
    Quantity,
    Item_Name,
    Price,
    Amount=Quantity*Price,
    Order_Date_Time,
    Cart_Status=1,
}

Status: {
    If successful: 201_Created
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

b. Student remove item from cart

```
URL: /Canteen/Student_Cart

Method: DELETE

Response: Delete item from the cart

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

c. Student confirms order

```
URL: /Canteen/Student_Cart

Method: POST

Data: {
    User_Name,
    Name,
    Quantity,
    Item_Name,
    Price,
    Amount=Quantity*Price,
    Order_Date_Time,
    Cart_Status=0,
    Processing_Status=1,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

d. Student view pending order

```
URL: /Canteen/Student_Pending_Order

Method: GET

Response: Display pending orders of the student

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

e. Student view orders history

```
URL: /Canteen/Student_Orders_History

Method: GET

Response: Display orders history of the student

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

f. Owner accept new order

```
URL: /Canteen/Owner_New_Order

Method: POST

Data: {
    User_Name,
    Name,
    Quantity,
    Item_Name,
    Price,
    Amount=Quantity*Price,
    Order_Date_Time,
    Processing_Status=0,
    Accepted_Status=1
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

g. Owner reject new order

```
URL: /Canteen/Owner_New_Order

Method: POST

Response: {
    Reject the order
    Accepted_Status=0,
    History_Status=1,
    Processing_Status=0
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

h. Owner view pending order

```
URL: /Canteen/Owner_Pending_Order

Method: GET

Response: Display pending orders to owner

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

i. Owner modify served status of item

```
URL: /Canteen/Owner_Pending_Order

Method: POST

Data: {
    User_Name,
    Name,
    Quantity,
    Item_Name,
    Price,
    Amount=Quantity*Price,
    Order_Date_Time,
    Served_Status=1
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

j. Owner modify payment status if item

```
URL: /Canteen/Owner_Pending_Order

Method: POST

Data: {
    User_Name,
    Name,
    Quantity,
    Item_Name,
    Price,
    Amount=Quantity*Price,
    Order_Date_Time,
    Payment_Status=1
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

k. Owner remove item from pending order

```
URL: /Canteen/Owner_Pending_Order

Method: POST

Response: {
    Delete item from pending order,
    History_Status=1
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

l. Owner add item

```
URL: /Canteen/Owner_Modify_Menu

Method: POST

Data: {
    Item_Name,
    Price,
}

Status: {
    If successful: 201_Created
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

m. Owner modify item

```
URL: /Canteen/Owner_Modify_Menu

Method: POST

Data: {
    Item_Name,
    Price,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

n. Owner delete item

```
URL: /Canteen/Owner_Modify_Menu

Method: POST

Response: Delete item from the menu

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

o. Owner view students' bill

```
URL: /Canteen/Owner_Students_Bill

Method: POST

Data: User_Name

Response: Display canteen bill of the student

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (User_Name is missing)
    }
}
```

3. Mess

a. Student view regular menu

```
URL: /Mess/Student_Regular_Menu

Method: POST

Response: Display regular menu to student

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

b. Student book extras

```
URL: /Mess/Student_Book_Extras

Method: POST

Data: {
    idt,
    Order_Date_Time,
    Meal_Date,
    User_Name,
    Item_Name,
    Price,
    Quantity,
    Amount,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

c. Student view booked extras

```
URL: /Mess/Student_Booked_Extras

Method: POST

Response: Display booked extras to student

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

d. Student apply for rebate

```
URL: /Mess/Student_Apply_For_Rebate

Method: POST

Data: {
    User_Name,
    from_date,
    to_date
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

e. Student view orders history

```
URL: /Mess/Student_Order_History

Method: POST

Response: Display orders history to student

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

f. Manager modify menu

```
URL: /Mess/Manager_Modify_Menu

Method: POST

Data: {
    idt,
    items
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

g. Manager add extras

```
URL: /Mess/Manager_Extra_Items

Method: POST

Data: {
    meal_date,
    meal,
    item,
    capacity,
    price,
    booking_date,
    start_time,
    end_time
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

h. Manager react on rebate requests

```
URL: /Mess/Manager_Rebate_Requests

Method: POST

Data: {
    username,
    name,
    fromdt,
    todt,
    rebate_days,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

i. Manager view students' bill

```
URL: /Mess/Manager_Students_Bills

Method: POST

Data: {
    User_Name,
    Bill_Month
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

j. Manager view orders history

```
URL: /Mess/Manager_View_Orders

Method: POST

Response: Display orders to manager

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

k. Manager view feedback

```
URL: /Mess/Manager_View_Feedback

Method: POST

Response: Display feedback to manager

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

l. Manager modify BDMR

```
URL: /Mess/Manager_Modify_BDMR

Method: POST

Data: {
    BDMR
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (BDMR is missing)
    }
}
```

4. Booking

a. Student book guestroom

```
URL: /Booking/guestroom

Method: POST

Data: {
    User_Name,
    Name,
    Date,
    Checkin_date,
    Checkout_date,
    Price
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

b. Student issue sports equipments

```
URL: /Booking/sports_equipments

Method: POST

Data: {
    User_Name,
    Name,
    Date,
    Equipment_selected,
    Secy_validation,
    Student_return_request
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

c. Student book sports court

```
URL: /Booking/courts

Method: POST

Data: {
    User_Name,
    Name,
    Date,
    Sports,
    Day_of_booking,
    Time_of_checkin,
    Time_of_checkout,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

d. Hall manager react on booking

```
URL: /Booking/booking_manager

Method: POST

Response: Approves or denies booking

Data: {
    User_Name,
    Name,
    Date,
    Sports,
    Day_of_booking,
    Time_of_checkin,
    Time_of_checkout,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

e. Secy equipment issue request validation

```
URL: /Booking/secy_request_validation

Method: POST

Response: Validates issuance of sports equipment

Data: {
    User_Name,
    Name,
    Date,
    Equipment_selected,
    Secy_validation,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

f. Secy equipment return request validation

```
URL: /Booking/secy_return_validation

Method: POST

Response: Validates returning of sports equipment

Data: {
    User_Name,
    Name,
    Date,
    Equipment_selected,
    Student_return_request,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

g. Secy add equipment

```
URL: /Booking/secy_add_equipment

Method: POST

Response: Student can issue an sports equipment

Data: {
    User_Name,
    Name,
    Date,
    Equipment_selected,
    Quantity
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

5. Cleaning

a. Student view pending request

```
URL: /Cleaning/Pending_Request

Method: POST

Response: View pending requests

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

b. Student view past request

```
URL: /Cleaning/Past_Request

Method: POST

Response: View past requests

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
    }
}
```

c. Student lodge request

```
URL: /Cleaning/Lodge_Request

Method: POST

Response: Student can lodge a new request

Data: {
    User_Name,
    Name,
    Room,
    Comments,
    Done
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

d. Hall manager view cleaning requests

```
URL: /Cleaning/Cleaning_hall

Method: POST

Data: {
    User_Name,
    Name,
    Done,
    Place,
    Comments,
    Cleaning_DateTime,
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (Some required data is missing)
    }
}
```

6. My Account

a. Student view mess bill

```
URL: /My_Account/Mess

Method: POST

Data: {
    username
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (username is missing)
    }
}
```

b. Student view canteen bill

```
URL: /My_Account/Canteen

Method: POST

Data: {
    username
}

Status: {
    If successful: 200_OK
    Else: {
        401_Unauthorized (Not logged in)
        400_Bad Request (username is missing)
    }
}
```

3 Completeness

SRS Features with status and their Future Plans

SRS features	Status	Future Development Plan
User Profile	Completed	Addition of Profile images and connecting the User Profile with Student Search.
Mess Automation	Completed	Incorporation of Calories intake of the Meal and Sale analysis of Extra's Item for Mess Manager.
Canteen Automation	Completed	Incorporation of Rating system and sale analysis for Canteen Owner.
Cleaning Automation	Completed	Display of Cleaning Staff Database for Hall and Notification system to Alert Cleaning Staff.
Booking Automation	Completed	Incorporation of Hall Facility available in future for Booking.
Student's Account	Completed	Incorporation of Electricity bills and Penalties.

Other Future Plans

- We are planning to expand our software for **Pan Campus**, including functionality like segregating users based on Hall, Accessing other Hall Canteen, etc.
- Introduction of a **Notification and Announcement** Section for the Hall.
- Enhancing the UI of the Portal.

Appendix A - Group Log

S.No.	Date	Duration	Venue	Description
1.	12/02/2023	120 min	RM Building	Distribution of work and Decided the utilities and libraries to use.
2.	15/02/2023	105 min	RM Building	Discussion on frontend progress.
3.	28/02/2023	70 min	RM Building	Compiling all the HTML and CSS together and discussing necessary changes.
4.	03/03/2023	120 min	RM Building	Discussion on updated HTML and CSS files and Work distribution of Backend.
5.	05/03/2023	100 min	RM Building	Discussion on initial models and views for Login, Signup Pages.
6.	06/03/2023	150 min	RM Building	Discussion on initial models and views for Mess and Canteen.
7.	07/03/2023	150 min	RM Building	Discussion on initial models and views for other Pages.
8.	08/03/2023	100 min	RM Building	Databasing of software.
9.	10/03/2023	120 min	RM Building	Discussion on final models and views for all Pages.
10.	11/03/2023	150 min	RM Building	Finalized overall software backend as well as frontend also modified to achieve better results.
11.	12/03/2023	180 min	RM Building	Checked on software functionality, performance, reliability, accessibility, efficiency, correctness, usability, integrity
12.	13/03/2023	90 min	RM Building	Locating and Fixing bugs in the software.
13.	14/03/2023	110 min	RM Building	Locating and Fixing bugs in the software.
14.	15/03/2023	55 min	RM Building	Worked on the Creation of Implementation Document.
15.	16/03/2023	70 min	RM Building	Discussion on changes for the Implementation Document to produce the first draft.
16.	19/03/2023	40 min	Hall-2	Discussion on final first draft of Implementation Document.