PES UNIVERSITY
BENGALURU
Department of Computer Science and Engineering

Deliverable 2: Planning, designing and implementation details(To submit on Sep 30, 2024)
Project Plan Document

PLANNING, DESIGN AND IMPLEMENTATION DOCUMENT

NAME 1: Sahana Bhat

SRN 1: PES1UG22CS500

SEC : 'I'

NAME 2: Shama Kiran

SRN 2: PES1UG22CS540

SEC : 'I'

1. Identify the Lifecycle to be Followed and Justification

We will be using the **Iterative Development Model** for the execution of this project. The main reason for choosing this model is its flexibility to allow incremental improvements and feature additions over multiple iterations. Since the system requires multiple features (such as login, assignments, messaging, etc.), building a basic functional system first and then gradually expanding it ensures that progress is visible, and feedback can be incorporated at each stage.

- **Advantages of Iterative Model for this Project**:
    - **Continuous Feedback**: Each iteration can be tested with users, ensuring the system meets requirements before adding more features.
    - **Risk Management**: Potential risks and bugs can be identified and fixed early with each iteration.

- **Scope for Change**: If new features or adjustments are needed, they can be added in subsequent iterations without significant disruption.
- **Initial Delivery**: We can deliver a working product early in the project, which is important for demonstrating progress.

---

2. Tools for Each Phase of the Lifecycle

- **Planning Tool**:
  - **Trello** to manage tasks and track progress, ensuring each iteration is clearly defined.
- **Design Tool**:
  - **Canva** for UI/UX design to create wireframes and mockups for each feature (login, assignment upload, etc.).
- **Version Control**:
  - **Git** with **GitHub** to manage code versions and enable collaboration. This will help in tracking changes over iterations and avoid conflicts.
- **Development Tool**:
  - **Visual Studio Code** for coding the website using **HTML, CSS, Python (Django)**, as it supports multiple languages and frameworks.
- **Bug Tracking**:
  - **Jira** to track bugs encountered during development, testing, and integration.
- **Testing Tool**:
  - **PyTest** for unit testing the Python/Django backend.

---

3. Deliverables and Reuse/Build Components

- **Deliverables**:

- **Reuse Components**:
  - **Django's Authentication System**: Instead of building the entire user login and registration from scratch, we can reuse Django's built-in user authentication and session management features.
  - **Django ORM**: Reuse the Django ORM for database interactions, ensuring we don't need to build a database layer manually.
- **Build Components**:
  - **Custom Features (Assignment Upload, Class Manager)**: These features will be built from scratch to meet the unique requirements of the system.
  - **Message System**: A custom message feature where students can send messages to teachers will also be built.

---

4. Work Breakdown Structure (WBS)

Below is the detailed breakdown of tasks for the project:

1. **Iteration 1: Basic System Setup**
   - **Task 1.1**: Project Initialization (Set up Django, create GitHub repo)
   - **Task 1.2**: User Login and Registration (Frontend and backend using Django's built-in features)
   - **Task 1.3**: Database Design (Define models for User, Class, Assignment, etc.)
2. **Iteration 2: Core Functionalities**
   - **Task 2.1**: Create Class Manager (Teachers add students, view class list)
   - **Task 2.2**: Upload Assignment (Teachers can upload assignments)
   - **Task 2.3**: View Assignment List (Students can view/download assignments)
3. **Iteration 3: Enhanced Features**

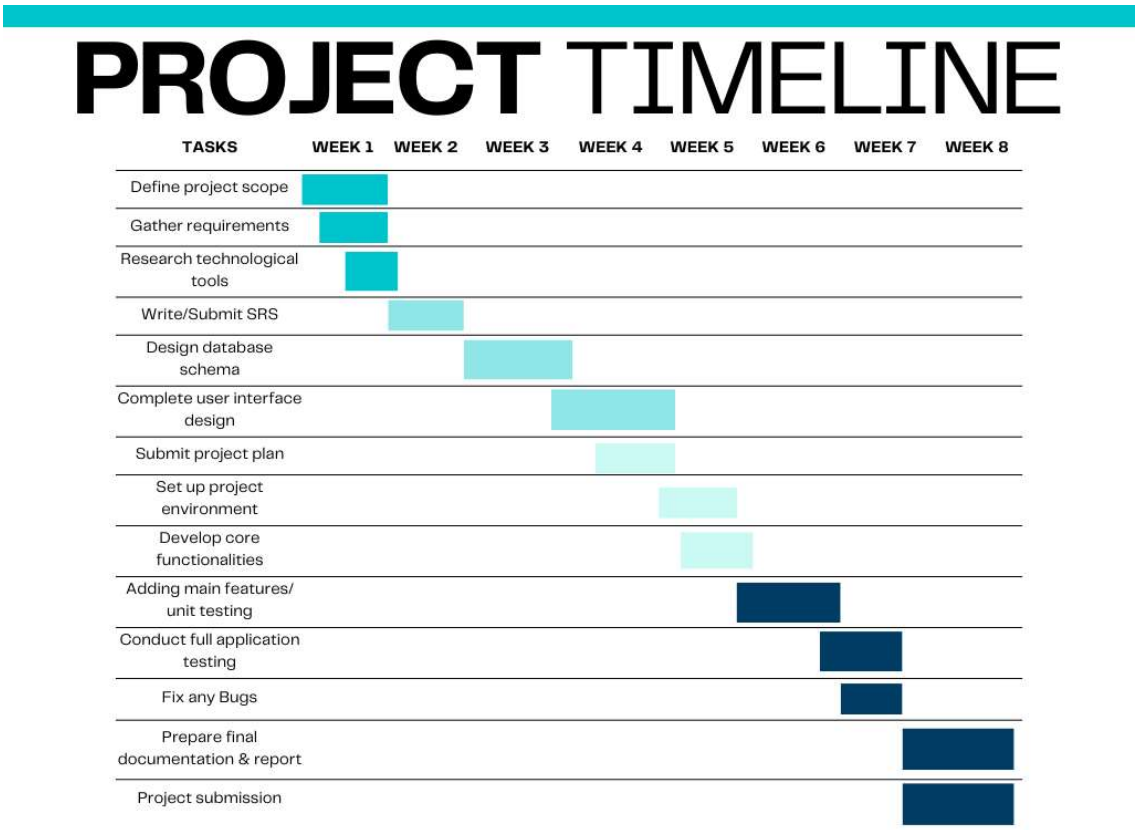
    - **Task 3.1**: Submit Assignment (Students can submit their assignments)
    - **Task 3.2**: View/Update Marks (Teachers can view/update student marks)
    - **Task 3.3**: Messaging System (Allow students to message teachers)
4. **Iteration 4: Additional Features and Testing**
    - **Task 4.1**: Notification System (Teachers can send notices to students)
    - **Task 4.2**: Search Functionality (Search students/teachers in the system)
    - **Task 4.3**: Testing (Run unit tests, integration tests, and fix bugs)
    - **Task 4.4**: Deployment (Deploy the system on a web server)

---

5. Effort Estimation and Gantt Chart

Below is a rough estimate of effort required for each task (in person-months):

- **Iteration 1**: 0.5 person-months (Setting up the system and basic login)
- **Iteration 2**: 1 person-month (Adding class manager and assignment features)
- **Iteration 3**: 1.5 person-months (Adding assignment submission and messaging system)
- **Iteration 4**: 0.5 person-months (Notification system, search, testing, and deployment)



---

6. Coding Details

Coding will follow an iterative process. After each feature is built, it will be thoroughly tested before moving to the next phase. Below is an overview of the coding approach:

- **Technologies**:
    - **Frontend**: HTML, CSS for basic structure and styling.
    - **Backend**: Python/Django for handling database operations, user authentication, and custom functionalities like assignment management.
    - **Database**: SQLite for development; it can be upgraded to PostgreSQL/MySQL for production.

6.1 Sample Code (models for user, teacher, student for the database)

```
from django.db import models

from django.contrib.auth.models import AbstractUser

from django.urls import reverse

from django.conf import settings

import misaka


class User(AbstractUser):

    is_student = models.BooleanField(default=False)

    is_teacher = models.BooleanField(default=False)


class Student(models.Model):

    user =
models.OneToOneField(User,on_delete=models.CASCADE,primary_key=True,related
_name='Student')

    name=models.CharField(max_length=250)

    roll_no = models.CharField(max_length=50)

    email = models.EmailField(max_length=254)

    phone = models.IntegerField()

    student_profile_pic =
models.ImageField(upload_to="classroom/student_profile_pic",blank=True)


    def get_absolute_url(self):

        return reverse('classroom:student_detail',kwargs={'pk':self.pk})


    def __str__(self):

        return self.name


    class Meta:

        ordering = ['roll_no']


class Teacher(models.Model):

    user =
models.OneToOneField(User,on_delete=models.CASCADE,primary_key=True,related
_name='Teacher')

    name = models.CharField(max_length=250)
```

```
    subject_name = models.CharField(max_length=250)

    email = models.EmailField(max_length=254)

    phone = models.IntegerField()

    teacher_profile_pic =
models.ImageField(upload_to="classroom/teacher_profile_pic",blank=True)

    class_students =
models.ManyToManyField(Student,through="StudentsInClass")


    def get_absolute_url(self):

        return reverse('classroom:teacher_detail',kwargs={'pk':self.pk})


    def __str__(self):

        return self.name


class StudentMarks(models.Model):

    teacher =
models.ForeignKey(Teacher,related_name='given_marks',on_delete=models.CASCA
DE)

    student =
models.ForeignKey(Student,related_name="marks",on_delete=models.CASCADE)

    subject_name = models.CharField(max_length=250)

    marks_obtained = models.IntegerField()

    maximum_marks = models.IntegerField()


    def __str__(self):

        return self.subject_name


class StudentsInClass(models.Model):

    teacher =
models.ForeignKey(Teacher,related_name="class_teacher",on_delete=models.CAS
CADE)

    student =
models.ForeignKey(Student,related_name="user_student_name",on_delete=models
.CASCADE)


    def __str__(self):

        return self.student.name
```

```
    class Meta:
        unique_together = ('teacher','student')


class MessageToTeacher(models.Model):
    student =
models.ForeignKey(Student,related_name='student',on_delete=models.CASCADE)
    teacher =
models.ForeignKey(Teacher,related_name='messages',on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now=True)
    message = models.TextField()
    message_html = models.TextField(editable=False)


    def __str__(self):
        return self.message


    def save(self,*args,**kwargs):
        self.message_html = misaka.html(self.message)
        super().save(*args,**kwargs)


    class Meta:
        ordering = ['-created_at']
        unique_together = ['student','message']


class ClassNotice(models.Model):
    teacher =
models.ForeignKey(Teacher,related_name='teacher',on_delete=models.CASCADE)
    students = models.ManyToManyField(Student,related_name='class_notice')
    created_at = models.DateTimeField(auto_now=True)
    message = models.TextField()
    message_html = models.TextField(editable=False)


    def __str__(self):
        return self.message


    def save(self,*args,**kwargs):
        self.message_html = misaka.html(self.message)
```

```
        super().save(*args,**kwargs)


    class Meta:

        ordering = ['-created_at']

        unique_together = ['teacher','message']


class ClassAssignment(models.Model):

    student =
models.ManyToManyField(Student,related_name='student_assignment')

    teacher =
models.ForeignKey(Teacher,related_name='teacher_assignment',on_delete=model
s.CASCADE)

    created_at = models.DateTimeField(auto_now=True)

    assignment_name = models.CharField(max_length=250)

    assignment = models.FileField(upload_to='assignments')


    def __str__(self):

        return self.assignment_name


    class Meta:

        ordering = ['-created_at']


class SubmitAssignment(models.Model):

    student =
models.ForeignKey(Student,related_name='student_submit',on_delete=models.CA
SCADE)

    teacher =
models.ForeignKey(Teacher,related_name='teacher_submit',on_delete=models.CA
SCADE)

    created_at = models.DateTimeField(auto_now=True)

    submitted_assignment =
models.ForeignKey(ClassAssignment,related_name='submission_for_assignment',
on_delete=models.CASCADE)

    submit = models.FileField(upload_to='Submission')


    def __str__(self):

        return "Submitted"+str(self.submitted_assignment.assignment_name)
```

```
class Meta:
    ordering = ['-created_at']
```