Chapter Four: Implementation

4.1 Implementation Overview

The implementation phase transforms the design specifications into functional code, utilizing the Java programming language within the Eclipse Integrated Development Environment (IDE). This stage focuses on developing the core classes identified in the class diagram, ensuring they align with the system's requirements and provide a robust foundation for task management. The process involves coding, testing, and deploying the application, with Visual Paradigm assisting in generating initial code skeletons.

4.2 Sample Class Implementations

User.java:

Class Diagram1 | TaskTest.java | AdminTest.java | NotificationTest.java | ReminderTest.java | SchedulerTest.java | UserTest.java | User.java

```java
package task_controller;

import java.util.List;

public class User {
    protected String name;
    protected String email;
    protected String password;
    protected String role;
    protected int user_id;

    public User(String name, String email, String password, String role, int user_id) {
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role;
        this.user_id = user_id;
    }

    public User() {
    }

    public boolean signUp(String name, String email, String password) {
        throw new UnsupportedOperationException("Sign-up not implemented yet.");
    }

    public boolean login(String email, String password) {
        throw new UnsupportedOperationException("Login not implemented yet.");
    }

    public void editProfile() {
            throw new UnsupportedOperationException("Edit profile not implemented yet.");
    }

    protected List<Task> searchTask(String keyword) {
        throw new UnsupportedOperationException("Search task not implemented yet.");
    }

    public List<Task> viewTasks() {
```

Task.java:

Class Diagram1 | *Task.java

```java
package task_controller;

import java.time.LocalDateTime;

public class Task {
    private int task_id;
    private int user_id;
    private String title;
    private String description;
    private LocalDateTime due_date;
    private String priority;
    private String status;
    private String category;

    public Task(int task_id, int user_id, String title, String description, LocalDateTime due_date, String priority,
                String status, String category, LocalDateTime created_at, LocalDateTime start_time, LocalDateTime end_tim
        this.task_id = task_id;
        this.user_id = user_id;
        this.title = title;
        this.description = description;
        this.due_date = due_date;
        this.priority = priority;
        this.status = status;
        this.category = category;
    }

    public boolean createTask(LocalDateTime due_date) {
        throw new UnsupportedOperationException();
    }

    public boolean editTask(int task_id, String title, String description, LocalDateTime due_date, String priority,
                            String status, String category) {
        throw new UnsupportedOperationException();
    }

    public boolean deleteTask(LocalDateTime start_time, LocalDateTime end_time) {
        throw new UnsupportedOperationException();
    }

    public Task distributeTask(int task_id) {
```

Writable | Smart Insert | 24 : 34 : 815

Reminder.java :

```java
package task_controller;

import java.sql.Date;

public class Reminder {
    private int reminder_id;
    private int task_id;
    private String status;
    private Date reminder_time;

    public Reminder(int reminder_id, int task_id, String status, Date reminder_time) {
        this.reminder_id = reminder_id;
        this.task_id = task_id;
        this.status = status;
        this.reminder_time = reminder_time;
    }

    public Reminder setReminder(int admin_id) {
        throw new UnsupportedOperationException();
    }

    public Reminder editReminder(int report_id) {
        throw new UnsupportedOperationException();
    }
}
```

4.3 Tools Used

- **IDE**: Eclipse, a versatile development environment with robust debugging and code generation features.
- **Language**: Java, chosen for its platform independence and extensive library support.
- **Other Tools**: Visual Paradigm, utilized for generating initial code skeletons from class diagrams and ensuring design-code alignment.

4.4 Deployment

The application is deployed using Eclipse run configurations, allowing execution in both console mode for testing and GUI mode for user interaction. The deployment process involves compiling the source code, running unit tests, and launching the application on a local server or standalone environment. Future enhancements may include deploying to a web server like Apache Tomcat to support broader access.