for

Software Requirements Specification

**Version 1.0**

Date created : 20th October 2024

*Department of Computer Science*

*Aligarh Muslim University*

Prepared by : *Mohd Saud*

23CAMSA107

GK0827

2024-2025

Automating Lab Management using GitHub

**LabTracker**

Table of Contents

**Table of Contents……………………………………………………………….………..…………….1**

[1. Introduction 2](#_Toc180348601)

[1.1 Purpose 2](#_Toc180348602)

[1.2 Document Convention 2](#_Toc180348603)

[1.3 Intended Audience and Reading Sugesstions 2](#_Toc180348604)

[1.4 Project Scope 3](#_Toc180348605)

[1.5 References 3](#_Toc180348606)

[2. Overall Description 3](#_Toc180348607)

[2.1 Product Perspective 3](#_Toc180348608)

[2.2 Product Features 4](#_Toc180348609)

[2.3 User Classes and Characteristics 5](#_Toc180348610)

[2.4 Operating Environment 6](#_Toc180348611)

[2.5 Design and Implementation Constraint 6](#_Toc180348612)

[2.6 User Documentation 7](#_Toc180348613)

[2.7 Assumption and Dependencies 7](#_Toc180348614)

[3. System Features 7](#_Toc180348615)

[3.1 Student Features 7](#_Toc180348616)

[3.2 Faculty Features 8](#_Toc180348617)

# 1. Introduction

## 1.1 Purpose

The purpose of **LabTracker** is to automate the lab management process for the Department of Computer Science, AMU. The system enables students to submit programming problems via GitHub and helps teachers track their progress efficiently. By integrating with the GitHub API, LabTracker provides teachers with insights into students' commitment patterns and allows for the generation of various reports to monitor class performance. It streamlines the workflow for teachers by automating tasks like problem submission tracking and deadline management.

## 1.2 Document Convention

This document uses the following conventions:

- **Bold**: Indicates key terms or concepts that are important within the context of the document.

- ***Italic*:** Used for emphasis or to denote specific titles, names of documents, applications, or sections.

- **List Items**: Bullet points are used to enumerate features, requirements, or important notes for clarity and easy reference.

- **Headings**: Major sections are marked with headings of different levels to structure the document and aid in navigation.

**- References**: URLs and external resources are provided in a consistent format for easy access and further reading.

## 1.3 Intended Audience and Reading Sugesstions

This document is intended for:

* **Project Stakeholders**: This includes faculty members and department heads who will oversee the implementation of LabTracker.
* **End Users**: Students and teachers who will interact with the system.

Reading Suggestions:

* Stakeholders should focus on sections detailing the project scope and system overview.
* End users might find the user interface descriptions and functionalities most relevant.

## 1.4 Project Scope

**LabTracker** is a web-based application aimed at automating lab management for the Department of Computer Science at AMU. It enables students to submit programming assignments via GitHub, allowing teachers to efficiently track progress and generate performance reports.

**Objectives:**

* **Streamline Submission Processes**: Simplify how students submit assignments.
* **Enhance Teacher Efficiency**: Reduce administrative tasks through automation.
* **Provide Analytics**: Offer insights into student performance to inform teaching strategies.

**Benefits:**

* **Time Savings**: Automation saves time for both students and faculty.
* **Improved Accountability**: Students can easily track deadlines and submissions.
* **Data Management**: Simplifies retrieval and analysis of performance data.

**LabTracker** aligns with the Computer Science Department's goals by integrating technology into education, preparing students for industry practices.

## 1.5 References

* IEEE 830-1998: IEEE Recommended Practice for Software Requirements Specifications.
* Django Documentation: <https://docs.djangoproject.com>
* GitHub API Documentation: <https://docs.github.com/en/rest>
* Project Management Institute (PMI): Standards and guidelines for project management practices.

# 2. Overall Description

## 2.1 Product Perspective

**LabTracker** is an enhancement to traditional lab management systems specifically tailored for the Department of Computer Science at AMU. While existing systems manage lab assignments and track submissions, they lack the ability to integrate directly with GitHub, which is a vital tool for modern software development.

**Context and Origin:**

* **Motivation**: With the rise of GitHub as a standard for code collaboration, a solution that automates submission tracking and assessment is essential.
* **Target Audience**: Designed for students and faculty in the Computer Science department, LabTracker streamlines the submission process and provides insights into student performance.
* **Unique Positioning**: LabTracker distinguishes itself by automating the retrieval of assignments and tracking progress via GitHub, improving efficiency and accuracy.

**Relation to Larger Systems:**

LabTracker serves as an integral component that enhances existing lab management systems through GitHub integration, connecting to:

* **GitHub**: For real-time submission tracking.
* **Existing Lab Management Systems**: Future versions may offer interoperability with other educational platforms.

## 2.2 Product Features

**LabTracker** provides several key features that enhance lab management by integrating with GitHub to automate tracking of student assignments. The major features are organized as follows:

**1. Student Management**

* **Student Registration**: Enables students to sign up and create profiles within the system.
* **GitHub Profile Validation**: Verifies the existence of a student’s GitHub account and checks for the necessary repositories related to their courses.

**2. Assignment Tracking**

* **GitHub Submission Monitoring**: Automatically tracks student submissions by monitoring their GitHub repositories for commit dates and activity related to assigned problems.
* **Completion Status Tracking**: Updates the completion status of assignments based on the last commit date from GitHub.

**3. Reporting and Analytics**

* **Individual Student Reports**: Generates detailed reports for faculties on individual student performance, including submission dates and problem completion statuses.
* **Class Performance Reports**: Provides aggregate analytics for the entire class, highlighting trends in submissions, on-time completions.
* **Faculty Activity Logs**: Maintains records of faculty actions within the system, such as problem creation and deadline settings.

**4. Faculty Management**

* **Assignment Creation**: Allows faculty members to create and manage programming problems, including descriptions and deadlines.
* **Deadline Management**: Enables faculty to set and update deadlines for assignments linked to specific problems.

## 2.3 User Classes and Characteristics

**1. Students**

* *Usage*: Regular (during lab sessions/assignments)
* *Functions*: Submit problems via GitHub, view progress, download reports, generate problem indexes.
* *Skills*: Basic GitHub and dashboard usage.
* *Privileges*: Access only their own data.
* *Importance*: High (primary users).

**2. Faculty**

* *Usage*: Frequent (for monitoring and evaluations).
* *Functions*: Track progress, generate reports, set deadlines, add problems, view activity logs.
* *Skills*: Moderate GitHub and web platform knowledge.
* *Privileges*: Access to all student data and reports.
* *Importance*: High (key managers of student progress).

**3. Administrators**

* *Usage:* Infrequent (for setup and troubleshooting).
* *Functions:* Manage users, maintain the system, handle GitHub API integration.
* *Skills:* High technical expertise (Django, APIs, databases).
* *Privileges:* Full system access.
* *Importance:* Moderate (support role).

## 2.4 Operating Environment

**1. Hardware Platform:**

* Runs on standard web servers (cloud-based or on-premises).
* Accessible via desktops, laptops, or mobile devices for end-users.

**2. Operating System:**

* *Server*: Linux (Ubuntu, Debian) or Windows Server.
* *Client*: Any OS (Windows, macOS, Linux, Android, iOS) with browser access.

**3. Software Components:**

* *Backend*: Django (Python framework).
* *Database*: PostgreSQL or MySQL.
* *Frontend*: HTML, CSS, JavaScript.
* *GitHub API*: For tracking student submissions and repositories.

**4. Dependencies:**

* Python 3.x and Django libraries.
* GitHub API access (requires API tokens).

**5. Browser Compatibility and Network Requirements:**

* Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.
* Reliable internet connection for GitHub API integration and web access.

LabTracker is designed to function smoothly in environments with minimal technical infrastructure, provided basic network access and web browser availability are ensured.

## 2.5 Design and Implementation Constraint

**1. Technologies and Tools :**

* Must use Django (backend) and GitHub API for submission tracking.
* PostgreSQL or MySQL as the database.

**2. API Rate Limit** : GitHub API limits real-time updates; only periodic tracking is possible.

**3. Security Requirements :**

* Authentication for students and teachers.
* Secure API tokens for GitHub integration.

**4. Maintenance Responsibility :** System must follow Django conventions to facilitate future maintenance by developers.

**5. Resource Constraints** **:** Efficient handling of GitHub data to avoid API and memory overloads.

## 2.6 User Documentation

**1. User Manual:**

* Detailed instructions for students and teachers on using the system.
* Includes screenshots and step-by-step guides.
* *Format*: PDF

**2. Quick Start Guide:**

* Summary of essential tasks (e.g., logging in, submitting problems, generating reports).
* *Format*: PDF and accessible via the homepage.

## 2.7 Assumption and Dependencies

**1. Assumptions :**

* Reliable internet access for users.
* GitHub API remains stable.
* Students understand GitHub usage.
* Faculty can use the web interface.
* Institution provides server and maintenance support.

**2. Dependencies :**

* GitHub API: For tracking submissions.
* Server Maintenance: Ensures system stability.

# 3. System Features

## 3.1 Student Features

**3.1.1 Progress Tracking**

*Description :* This feature allows students to track their progress on assignments by monitoring submissions made to their GitHub repositories.

*Priority :* High

*Stimulus/Response Sequences :*

* *Stimulus* : Student pushes code to GitHub.

*Response* : System updates the student’s progress on their dashboard.

*Functional Requirements* :

* *REQ-1* : Connect to GitHub API to track student submissions.
* *REQ-2* : Display completed, pending and late submissions on the student dashboard.

**3.1.2 File and Index Generation**

*Description :* Students can generate structured submission files and download indexes.

*Priority :* High

*Stimulus/Response Sequences :*

* *Stimulus* : Student requests index generation.

*Response* : System generates a DOCX file with problem details and images.

* *Stimulus* : Student generates a file for a specific week’s work.

*Response* : System creates a document with relevant problem information.

*Functional Requirements* :

* *REQ-1* : Allow students to generate DOCX files for lab submissions.
* *REQ-2* : Include problem descriptions, solutions and output images in generated files.
* *REQ*-3 : Provide downloadable indexes with formatting for teacher signatures.

## 3.2 Faculty Features

**3.2.1 Weekly Assignment Deadline Management**

*Description :* This feature allows teachers to set and manage deadlines for assignments, updating the deadlines displayed on students’ dashboards.

*Priority :* High

*Stimulus/Response Sequences :*

* *Stimulus* : Faculty sets or updates a deadline.

*Response* : System logs the change and updates the deadline on the student dashboard.

* *Stimulus* : Deadline is reached.

*Response* : System flags unsubmitted tasks as late in the reports.

*Functional Requirements* :

* *REQ-1* : Allow faculty to set and modify deadlines for weekly assignments.
* *REQ-2* : Automatically update the deadlines on students’ dashboards when changes are made.
* *REQ*-3 : Automatically mark overdue tasks as late and display them on the dashboard.

**3.2.2 Class Performance Monitoring and Reporting**

*Description :* Faculty can monitor class performance through detailed reports that track student progress and engagement. The system supports three types of reports:

* *Specific Student Report* : Provides insights into an individual student’s progress and submission history.
* *Class Weekly Report* : Summarizes weekly performance trends across the entire class.
* *Whole Class Report* : Offers an overview of the overall progress of the class across all weeks.

*Priority :* High

*Stimulus/Response Sequences :*

* *Stimulus* : Faculty requests a Specific Student Report.

*Response* : System generates a detailed report for the selected student.

* *Stimulus* : Faculty requests a Class Weekly Report

*Response* : System generated a detailed report for the class for the specified week.

* *Stimulus* : Faculty requests a Whole Class Report

*Response* : System generated a detailed report for the class across all weeks.

*Functional Requirements* :

* *REQ-1* : Generates Specific Student Reports showing individual student progress.
* *REQ-2* : Provide Class Weekly Reports summarizing weekly performance.
* *REQ*-3 : Generate Whole Class Reports showing overall class progress.
* *REQ*-4 : Ensure that all reports are secure and accessible only to authorized faculties.

**3.2.3 Teacher Activity Logs**

*Description :* The system tracks all actions performed by teachers, such as setting deadlines or adding problems, to maintain transparency and accountability.

*Priority :* Medium

*Stimulus/Response Sequences :*

* *Stimulus* : Faculty sets or modifies a deadline.

*Response* : System logs the action in faculty activity log.

* *Stimulus* : Faculty adds or edits a problem.

*Response* : System logs the action in faculty activity log.

* *Stimulus* : Faculty requests a report.

*Response* : System logs the action in faculty activity log.

* *Stimulus* : Faculty starts a new semester / updates the student data from GitHub.

*Response* : System logs the action in faculty activity log.

*Functional Requirements* :

* *REQ-1* : Log activities such as setting deadlines, adding, or editing problems.
* *REQ-2* : Record request for class-wide or weekly reports, starting a new semester, or updating the student data from GitHub.
* *REQ*-3 : Ensure only authorized teachers can access these logs.

**3.2.4 Adding and Editing Problems**

*Description :* Teachers can add new problems for students to solve and update existing ones. Problems are categorized by course, semester, and week to ensure proper tracking and accessibility.

*Priority :* High

*Stimulus/Response Sequences :*

* *Stimulus* : Faculty sets or modifies a deadline.

*Response* : System logs the action in faculty activity log.

* *Stimulus* : Faculty adds or edits a problem.

*Response* : System logs the action in faculty activity log.

* *Stimulus* : Faculty requests a report.

*Response* : System logs the action in faculty activity log.

* *Stimulus* : Faculty starts a new semester / updates the student data from GitHub.

*Response* : System logs the action in faculty activity log.

*Functional Requirements* :

* *REQ-1* : Log activities such as setting deadlines, adding, or editing problems.
* *REQ-2* : Record request for class-wide or weekly reports, starting a new semester, or updating the student data from GitHub.
* *REQ*-3 : Ensure only authorized teachers can access these logs.