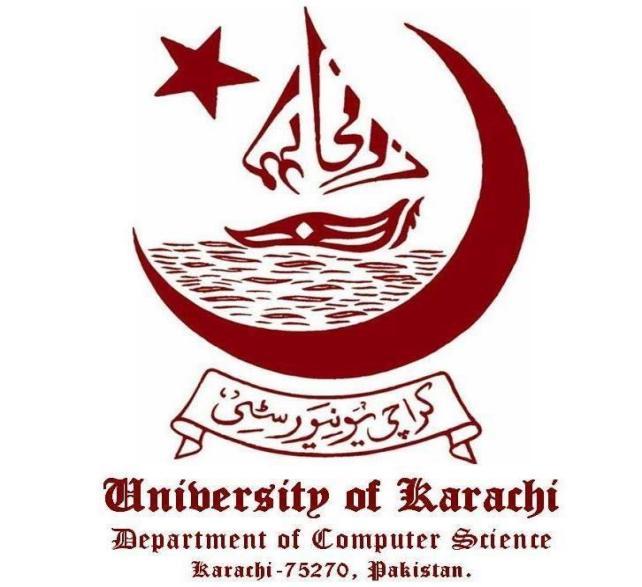
**CheckMate**

Say No to Manual Grading!



SUPERVISORS

Dr Shaista Raees & Dr Sadiq Ali Khan

**SUBMITTED BY**

HAMMAD FAHEEM EB21102029

HASSAN QAMAR EB21102037

MUHAMMAD UMER SALEEM EB20102098

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF KARACHI



**CheckMate**

Say No to Manual Grading!

18.02.2025

**Session: BSCS 2021 - 2025 (Evening)**

Submitted By:

HAMMAD FAHEEM EB21102029

HASSAN QAMAR EB21102037

MUHAMMAD UMER SALEEM EB20102098

Table Of Content

[**ABSTRACT 6**](#_dgnvkvsjeow4)

[SUMMARY 6](#_w43wdu6rypp2)

[Procedure 6](#_t0no772tn7e3)

[**Chapter1: INTRODUCTION 9**](#_77zpw18qa7i6)

[1.1 Overview 9](#_me87dnl11gy7)

[1.1.1 Objectives 9](#_q0sb9trmsrgw)

[1.1.2 Key Features 10](#_7wknib8d457)

[1.1.3 Technical Stack 10](#_pwn05d7bqjci)

[1.2 Functionalities Of Talent Scan 11](#_q8y2dpz4m8dy)

[1.3 Non-Functionalities 13](#_eksx9q3qpdnx)

[1.4 User Access Rights and Functionalities 14](#_ot38f5tt9j8o)

[1.4.1 User Dashboard Functionalities 14](#_jo22ypzg7ca)

[1.4.2 Company Dashboard Functionalities 14](#_1ailxtucemdr)

[1.5 Module Workflow 15](#_r1ju117n0vi7)

[1.5.1 Management Module Workflow 15](#_hxxg6416sdwe)

[1.5.2 Employee Monitoring and Reporting Module Workflow 15](#_8wqnj6khfrko)

[**Chapter 2: MOTIVATION FOR PROJECT 16**](#_tqt213326drp)

[2.1 Challenges in Traditional Recruitment Processes 16](#_2663lm57rjiz)

[2.1.1 Inefficiency in Manual Processes 16](#_2aoeyluqosb)

[2.1.2 Human Bias in Hiring Decisions 16](#_c6pta4aifdds)

[2.1.3 Scalability Issues 16](#_y6o8sti7hrg4)

[2.1.4 Lack of Personalization for Job Seekers 17](#_u66217jsfmxs)

[2.2 Opportunities for Innovation 17](#_551d8zhphd5l)

[2.2.1 Automation of Repetitive Tasks 17](#_hn0cie1kedr5)

[2.2.2 Data-Driven Decision Making 17](#_mrs1x3bq7qbj)

[2.2.3 Scalable and Adaptive Solutions 17](#_49vvbpaa2ua)

[2.2.4 Enhanced Candidate Experience 17](#_b6ykchnf9rv4)

[2.3 Addressing Industry Pain Points 18](#_eo6bcmrodw58)

[2.3.1 Reducing Time-to-Hire 18](#_ts5d0foxlo4f)

[2.3.2 Promoting Diversity and Inclusion 18](#_aljozy1vtt6x)

[2.3.3 Improving Recruitment Efficiency 18](#_75ug82mn2wou)

[2.3.4 Empowering Job Seekers 18](#_k00zeko5xw03)

[2.4 Vision for the Future of Recruitment 19](#_yt59h5d54f7g)

[**Chapter 3: REQUIREMENTS & SPECIFICATIONS 20**](#_8iqudluhb06n)

[3.1 Goals 20](#_pk3s55179yd9)

[3.2 Context 20](#_jlpq7tw29fbu)

[3.2.1 Industry Context 20](#_5xr86fnwhlpw)

[3.2.3 Technological Context 21](#_w017iciuy1td)

[3.3 Usage Scenarios 21](#_yqaibfsdj7d8)

[3.3.1 Job Seeker Scenario 21](#_gw3j3hkxty03)

[3.3.2 Employer Scenario 22](#_ibkj7we301w8)

[3.4 Special Usage Considerations 22](#_cfvuio8tlx3j)

[3.4.1 International Language 22](#_f860z0dmwfav)

[3.4.2 Data Privacy and Security 22](#_7hhmvnouy8w)

[3.4.3 Scalability 22](#_lizdrika0yc3)

[3.4.4 Accessibility 22](#_yei66b7vpi6e)

[3.5.1 Job Seekers 23](#_fvuei28m0as)

[3.5.2 Employers 23](#_dcc7yk4ri4po)

[3.5.3 HR Teams 23](#_d2gonpxzw91e)

[3.6 Risk Assessment 23](#_uce49d2b0rko)

[3.6.1 Technical Risks 23](#_ggy64u8c6ete)

[3.6.2 Operational Risks 24](#_jh123wf7fnvo)

[3.6.3 Compliance Risks 24](#_88qv0t1khsx9)

[3.6.4 Mitigation Strategies 24](#_nsqmtxvt61j0)

[**Chapter 4: ANALYSIS AND DESIGN 25**](#_kff18os3m89)

[4.1 System Architecture 25](#_8pr1w1fphxak)

[4.1.1 High-Level Architecture 25](#_92hh7vz27ins)

[4.1.2 Component Diagram 25](#_truw4drrh9qb)

[4.2 Design Principles 26](#_af9gp4m5q8iy)

[4.2.1 Modularity 26](#_6v8o1xqm02ed)

[4.2.2 Scalability 26](#_j730ptx02k9e)

[4.2.3 User-Centric Design 26](#_dsxgm9sp54li)

[4.2.4 Security 26](#_s3a0oyt1nn6f)

[4.3 Data Flow and Workflow 26](#_5nkspi1iqudv)

[4.3.1 Job Seeker Workflow 26](#_wbibovi04m9w)

[4.3.2 Employer Workflow 27](#_3zryg3vb2xl)

[4.3.3 Data Flow Diagram 27](#_i2mb18ke614n)

[4.4 Database Design 27](#_38twm3cevug5)

[4.4.1 Database Schema 27](#_o5lwvnkd9k3)

[4.4.2 Data Relationships 27](#_s9zic2wukiai)

[4.5 Prototype and Wireframes 28](#_4au57i2unoru)

[4.5.1 Job Seeker Dashboard 28](#_ix6qjqnef3b4)

[4.5.2 Company Dashboard 28](#_ysq2gjqpf0n8)

[4.6 Non-Functional Design Considerations 28](#_u50o8gcqq1gw)

[**Chapter 5: ALGORITHM DESCRIPTION FOR RECRUITMENT SYSTEM API 29**](#_11lzyoq67ybk)

[5.1 Overview of the Recruitment System API 29](#_4z2e5hgut4hx)

[5.2 Input Processing 29](#_qko112gbqdb)

[5.2.1 Input Types 29](#_4s5ff79emqn9)

[5.2.2 Preprocessing Steps 30](#_8ea7vbzbegoy)

[5.3 LLM Processing (Groq AI) 30](#_1ufxu7ffdii2)

[5.3.1 Input Submission 30](#_tc2bis73a16q)

[5.3.2 Key Tasks Performed by the LLM 30](#_drtjvq7z3ntx)

[5.4 Output Generation 31](#_pa68pg3erudh)

[5.4.1 Structured Responses 31](#_qhhhua4kvohf)

[5.4.2 Post-Processing 31](#_vzw3vldeboj2)

[5.4.3 Final Output 31](#_m7kpg9qq6f3u)

[5.5 Post-Processing & Integration 32](#_sgy0qi4z7n9a)

[5.5.1 Additional Logic 32](#_mer38q9cc91d)

[5.5.2 Integration with External Services 32](#_9q3516s35ivj)

[5.6 Benefits of the Algorithm 32](#_m421t6cu2rz4)

[**CHAPTER 6: DEVELOPMENT OF MATRICES, SKILL ASSESSMENT & GAP ANALYSIS 33**](#_lii9w4b39g9q)

[6.1 Development of Matrices 33](#_4u842bjb93kg)

[6.1.1 Purpose of Matrices 33](#_lzt1znywpjol)

[6.1.2 Types of Matrices 33](#_irbnuuf8kw7t)

[6.1.3 Matrix Development Process 34](#_fwcasksybuts)

[6.2 Skill Assessment Framework 34](#_z7odqd5vii5i)

[6.2.1 Importance of Skill Assessment 34](#_kdforlq5gzjx)

[6.2.2 Skill Assessment Methodology 34](#_1wqkhzbherkl)

[6.3 Gap Analysis 35](#_7q3wpkgky6bb)

[6.3.1 Purpose of Gap Analysis 35](#_wji7kekcfv8q)

[6.3.2 Gap Analysis Methodology 35](#_pwd2929n4ct)

[6.3.3 Gap Analysis Tools 35](#_rz39imwv7sol)

[6.4 Integration with Talent Scan Platform 36](#_kjp5h6ysnsy)

[6.4.1 Skill Assessment in Candidate Matching 36](#_fdok2gejhejf)

[6.4.2 Gap Analysis in Job Recommendations 36](#_3cq1efo5t8fy)

[6.4.3 Reporting and Analytics 36](#_i53u9qafoi56)

[6.5 Benefits of Matrices, Skill Assessment, and Gap Analysis 37](#_656ynil7xuf2)

[6.5.1 For Organizations 37](#_7frgf9nh2p5k)

[6.5.2 For Job Seekers 37](#_syte88196nu)

[6.6 Case Study: Implementing Skill Assessment and Gap Analysis 37](#_5e70nqv2wsf9)

[6.6.1 Scenario 37](#_ju1w6kzbykww)

[6.6.2 Process 37](#_2y5sldi7wk9z)

[6.6.3 Outcome 38](#_qzbia0v7jqx0)

[**CHAPTER 7: REFERENCES 39**](#_l3xg5qffyh2t)

[7.1 Web and Mobile Application Design 39](#_tor6lrwxhuz)

[7.2 Miscellaneous References 40](#_iof8dkz2ugn9)

[**Final Thoughts 41**](#_hmldup11qa3q)

[List Of Figures: 42](#_bieh33zdkk30)

[ERD DIAGRAM 42](#_zgvi7iyb5s7w)

[**Process Flow Diagram 43**](#_23ako4wlxejp)

# **ABSTRACT**

### **SUMMARY**

The rapid advancement of Artificial Intelligence (AI) has transformed multiple domains, including education. This project proposes the development of **CheckMate – The AI-Powered Student Portal**, an intelligent system designed to automate and streamline the student assignment evaluation process. By leveraging AI-driven techniques such as Natural Language Processing (NLP) and Machine Learning (ML), the system evaluates student assignments, provides automated grading, and generates personalized feedback.

CheckMate addresses the inefficiencies of traditional manual grading, which is time-consuming, prone to subjectivity, and lacks scalability. The platform ensures consistent, unbiased, and data-driven assessment, allowing educators to focus on student engagement and learning. CheckMate is designed to handle high volumes of student submissions while adapting to evolving educational requirements.

### **Procedure**

### **System Architecture**

CheckMate is built on a **modern and scalable technology stack** to ensure efficiency and flexibility. The backend is developed using **NodeJs (Typescript)** for high-performance API handling, while the frontend is designed using **React.js** to provide a seamless user experience. For data storage, the system utilizes **MongoDB** to manage student submissions, evaluation results, and performance analytics.

### **AI and NLP Integration**

The project utilizes **Artificial Intelligence (AI) and Natural Language Processing (NLP)** to analyze, evaluate, and grade student assignments. The integration is achieved through:

#### **2.1 AI-Powered Assignment Evaluation using LLMs**

The system sends extracted text from student submissions to a **Large Language Model (LLM)** via an API request (e.g., OpenAI’s GPT, Google’s Gemini, or a locally hosted model). The model is instructed to assess the content based on predefined rubrics, checking for structure, coherence, relevance, and originality. The AI-generated assessment is structured into a **JSON-compatible format** for further processing.

#### **2.2 NLP-Based Text Extraction & Analysis**

Using libraries such as **spaCy** and **NLTK**, the system extracts and preprocesses student assignment text to ensure accurate evaluation. Features like **keyword detection, grammar analysis, and semantic similarity** enhance the assessment process.

#### **2.3 Automated Grading and Feedback Generation**

The AI model **evaluates student assignments** based on **grading criteria** defined by educators, assigning scores and providing **detailed feedback** on strengths and areas for improvement.

#### **2.4 Adaptive Learning Insights & Performance Analytics**

By analyzing student performance trends, the system **identifies learning gaps** and provides **personalized recommendations**, helping students improve over time.

### **Data Processing and Student Performance Evaluation**

The system processes assignments by first parsing the text using **NLP techniques**. It then applies **AI algorithms** to grade submissions, generate **automated feedback**, and offer **performance tracking** for students. The ranking is based on a combination of factors, including **content accuracy, depth of analysis, writing quality, and relevance to the assignment topic**. This automated approach ensures a **fair, unbiased, and efficient evaluation process**.

### **User Experience and Automation**

The frontend, built with **React.js**, provides an **interactive and user-friendly interface** for students and educators. The platform automates **grading, feedback generation, plagiarism detection, and performance tracking**, allowing educators to focus on personalized teaching strategies. Students benefit from **instant feedback and AI-driven insights** to enhance their learning experience.

# 

# 

# 

# **1. Introduction**

## **1.1 Overview**

CheckMate is an AI-powered student portal designed to **automate assignment evaluation, provide structured feedback, and track student performance**. The system utilizes **Natural Language Processing (NLP) and Machine Learning (ML)** to ensure efficient, unbiased, and data-driven assessment. By reducing the manual workload for instructors and delivering personalized insights to students, CheckMate enhances the learning experience.

## **1.1.1 Objectives**

The primary objectives of CheckMate include:

* **Automating assignment evaluation** to save time and reduce human bias.
* **Providing real-time feedback** to help students improve their work.
* **Enhancing student engagement** through performance analytics and insights.
* **Enabling instructors to track student progress** efficiently.
* **Ensuring a seamless, user-friendly experience** for students and instructors.

## **1.1.2 Key Features**

CheckMate offers a variety of features to **simplify assignment evaluation**:

* **AI-Powered Grading:** Automatically evaluates assignments based on instructor-defined criteria.
* **Instant Feedback Generation:** Provides students with **detailed improvement suggestions**.
* **Performance Tracking Dashboard:** Displays **student progress and historical performance trends**.
* **Customizable Evaluation Criteria:** Instructors can **define grading rubrics** based on assignment needs.
* **Multi-Format Support:** Accepts assignments in various formats (**PDF, Word, images, and code files**).
* **Seamless LMS Integration:** Can integrate with **Moodle, Google Classroom, or Blackboard**.

## **1.1.3 Technical Stack**

CheckMate is built using a **scalable and efficient tech stack**:

* **Frontend:** React.js (for a dynamic and responsive UI).
* **Backend:** FastAPI (a high-performance Python web framework).
* **Database:** PostgreSQL or MongoDB (for structured and flexible data storage).
* **AI/NLP Models:** GPT, BERT, or RoBERTa (for text analysis and feedback).
* **Cloud Storage:** AWS S3 or Firebase (for storing assignment files).
* **Authentication:** OAuth 2.0 and JWT (for secure user login).

# **2. System Functionalities**

## **2.1 Functionalities of CheckMate**

CheckMate simplifies **assignment submission, evaluation, and feedback** through:

* **Assignment Submission Portal:** Students upload assignments easily.
* **Automated Grading System:** AI analyzes and grades assignments based on instructor-set rubrics.
* **Feedback Mechanism:** Provides **real-time suggestions** to help students refine their work.
* **Instructor Dashboard:** Teachers manage assignments, grading, and student performance.
* **Student Dashboard:** Displays grades, feedback, and submission history.
* **Collaboration Tools:** Enables discussion forums between students and instructors.

## **2.2 Non-Functional Requirements**

CheckMate also ensures **performance, security, and scalability**:

* **High Availability:** The system remains accessible with minimal downtime.
* **Data Security & Privacy:** Protects user data through encryption and secure authentication.
* **Scalability:** Supports multiple institutions and large student bases.
* **User-Friendly Interface:** Designed for intuitive use across all devices.
* **Fast Response Time:** Processes assignments efficiently without delays.

# **3. User Access Rights and Functionalities**

## **3.1 Student Dashboard Functionalities**

The **Student Dashboard** allows users to:

* **Submit Assignments** and track submission deadlines.
* **View AI-Generated Feedback** for self-improvement.
* **Check Grades & Performance Trends** over time.
* **Access Detailed Reports** on their strengths and weaknesses.
* **Engage in Discussions** with instructors for additional clarification.

## **3.2 Instructor Dashboard Functionalities**

The **Instructor Dashboard** provides tools for effective evaluation:

* **Create & Manage Assignments** with custom grading rubrics.
* **Automated & Manual Grading Options** to adjust AI-evaluated scores if needed.
* **Detailed Student Performance Insights** to track class progress.
* **Customizable Feedback Options** to personalize AI-generated suggestions.
* **Download & Export Reports** for offline review and academic records.

# **4. Module Workflow**

## **4.1 Assignment Evaluation Workflow**

The **Assignment Evaluation Workflow** follows these steps:

1. **Students Upload Assignments** through the portal.
2. **AI Processes the Submission**, extracting key details.
3. **Automated Grading Takes Place** using pre-defined rubrics.
4. **Feedback is Generated & Displayed** to the student.
5. **Instructors Review & Adjust Scores** (if necessary).
6. **Final Results are Published** on student and instructor dashboards.

## **4.2 Performance Analytics and Reporting Workflow**

This workflow helps instructors **track student progress and class performance**:

1. **Data Collection:** The system gathers data on student performance.
2. **Trend Analysis:** AI identifies **patterns and common errors**.
3. **Automated Reports Generation:** Provides **performance insights**.
4. **Instructor Customization:** Teachers can **filter and analyze data** based on specific criteria.
5. **Recommendations for Improvement:** AI suggests learning strategies for students.
6. **Export & Sharing Options:** Reports can be downloaded for academic review.

# **AI Module of Checkmate App – Student Assignment Evaluation System**

The **Checkmate App** is designed to automate the evaluation of student assignments using an AI-powered system. The AI module comprises a series of specialized agents that process, evaluate, and provide feedback on student assignments. The system ensures accuracy, fairness, and efficiency in grading by leveraging **Natural Language Processing (NLP)** and **Large Language Models (LLMs)**.

### **Overview of the AI Process**

The AI module consists of four key agents:

1. **Formatting Agent**
2. **Rubric Agent**
3. **Evaluation Agent**
4. **Feedback Generation**

Each agent plays a crucial role in structuring, evaluating, and providing insights into student responses. The following sections detail the functionality and importance of each agent.

## **1. Formatting Agent**

The **Formatting Agent** is responsible for preprocessing student and teacher assignment data. Since assignments are submitted in various formats, such as **PDFs, DOC files, or plain text**, this agent extracts and structures the content into a standardized format.

### **Functionality**

* Reads and processes raw text from student and teacher assignments.
* Cleans unnecessary content such as metadata, page numbers, headers, and footers.
* Converts the extracted text into a **JSON format** consisting of a list of dictionaries, where:
  + Each dictionary contains a **question** and its corresponding **answer**.
* Ensures consistency in formatting for smooth downstream processing.

### **Significance**

The **Formatting Agent** is a critical component because unstructured text can lead to inaccurate evaluation results. By structuring the assignment into a standardized JSON format, the system ensures that every question and answer is correctly processed for evaluation.

## **2. Rubric Agent**

The **Rubric Agent** is responsible for defining the grading criteria for each question in an assignment. It ensures that marks are distributed fairly based on the complexity and importance of each question.

### **Functionality**

* Accepts the following inputs:
  + **Student JSON** (formatted assignment response from the student).
  + **Teacher JSON** (formatted assignment with correct answers).
  + **Total marks** assigned for the entire assignment.
* Dynamically assigns marks to each question based on:
  + The context and length of the question.
  + The difficulty level inferred from the teacher’s answer.
* Generates a **Rubric JSON** that contains:
  + Each **question** and its corresponding **allocated marks**.

### **Significance**

The **Rubric Agent** ensures that grading is fair and consistent. Instead of a fixed mark distribution, the agent adapts to the assignment’s structure and assigns marks proportionally, improving accuracy in evaluation.

## **3. Evaluation Agent**

The **Evaluation Agent** is the core AI model that assesses the student's responses against the teacher’s reference answers using the **rubric-based grading system**.

### **Functionality**

* Takes **three inputs**:
  + **Student JSON** (student's answers).
  + **Teacher JSON** (ideal answers).
  + **Rubric JSON** (marks distribution).
* Compares each student’s response with the expected answer based on:
  + **Relevance** – How well the student’s response aligns with the teacher’s answer.
  + **Completeness** – Whether the response sufficiently covers key concepts.
  + **Accuracy** – The correctness of information provided.
  + **Language & Clarity** – The coherence and grammatical correctness of the response.
* Assigns marks to each question based on **predefined rubric criteria**.
* Provides insights into the evaluation process by explaining how the AI model assessed each response.

### **Significance**

The **Evaluation Agent** ensures transparency in grading by making the assessment process visible. It does not just provide scores but also justifies why specific marks were given, enhancing trust in the AI-driven evaluation.

## **4. Feedback Generation**

The final step in the evaluation process is the generation of **personalized feedback** for students. The system aims not only to assign marks but also to guide students in improving their performance.

### **Functionality**

* Provides **question-specific feedback** highlighting strengths and areas for improvement.
* Suggests **key concepts** the student might have missed.
* Recommends **learning resources** or study techniques based on the student’s mistakes.
* Offers **general feedback** on writing clarity, coherence, and structuring answers effectively.

### **Significance**

By integrating **personalized feedback**, the **Checkmate App** helps students improve their understanding and performance in future assignments. This feature transforms the system from a mere grading tool into an **educational assistant**.

## **User Interface & Deployment**

Currently, the **Checkmate App** is implemented using **Streamlit**, which provides a simple and interactive user interface to display evaluation results. Users can upload assignments and view their scores, along with detailed explanations and feedback.

### **Future Plans**

* **Integration with Student Portal** – The system will be embedded into an existing **student portal**, allowing direct submission and evaluation of assignments.
* **Enhanced AI Capabilities** – Future iterations will include:
  + **Plagiarism Detection** to ensure originality.
  + **More Advanced NLP Models** for deeper understanding and better feedback.
  + **Multilingual Support** for assignments in various languages.

## **Conclusion**

The **Checkmate App** revolutionizes assignment evaluation by leveraging **AI-driven grading**. With a structured approach involving **formatting, rubric-based grading, evaluation, and feedback**, the system ensures a **fair, transparent, and insightful** assessment process. By automating grading while providing **human-like explanations**, the application enhances learning outcomes for students and reduces grading workload for educators.