HuBiCEM: A Prediction-Based Individual Developer Performance and Cost Estimation Model for Agile Development

Abdullah Ahmad[*], **Abdullah Bajwa**[**]

[*] Department of Artificial Intelligence, UMT Lahore

Abstract- Agile has been developed to enhance efficiency and address the challenges in software development. Today, it is widely adopted due to its ability to support both developers and clients effectively. Agile methodology fosters strong interaction between developers and clients, ensuring a high-quality product. One of its key features is the flexibility to accommodate changes at any stage of the project, allowing product owners to refine requirements. However, this very feature leads to increased project costs and delays due to frequent change requests. Traditional cost estimation techniques, such as COCOMO, Function Point Analysis, and expert judgment, struggle to provide accurate estimates in agile environments. To address this, we propose a new cost estimation model, HuBiCEM (Hunar Bin Cost Evaluation Model), which individually tracks developer performance rather than relying on a team-based assessment. Unlike conventional models, HuBiCEM evaluates multiple performance factors beyond time estimation and applies Linear Regression to predict future performance in upcoming sprints. We implemented the model a case study — an OpenAI Assistant Project— and conducted statistical validation using industry data. Our survey session with development teams aimed to assess the reliability and accuracy of the proposed model. The study is also statistically analyzing the effectiveness of HuBiCEM over traditional estimation techniques. The results demonstrate that HuBiCEM provides more precise cost and time estimations, making it a suitable model for agile software development.

Keywords- HuBiCEM Model, Individual Performance Estimation, Cost Prediction, Agile Cost Estimation, Polynomial Regression

I. Introduction

Agile software development offers flexibility by allowing changes at any stage, ensuring better alignment with client needs. However, this adaptability creates challenges in cost and time estimation, as frequent modifications lead to project delays and budget overruns. Traditional estimation models like COCOMO and Function Point Analysis focus on team-based assessments, failing to account for individual developer contributions and variations in expertise.

To address this gap, we propose HuBiCEM (Hunar Bin Cost Evaluation Model), which evaluates individual developer performance rather than relying on team-wide estimates. HuBiCEM tracks multiple performance factors and applies Linear Regression to predict future efficiency in upcoming sprints. We implemented this model in two case studies and validated its effectiveness through industry data and surveys. The results confirm that HuBiCEM improves estimation accuracy, making it a more suitable approach for agile software development.

II. Significance of Study

Frequent change requests in agile development enhance client satisfaction by allowing continuous refinement of project requirements. However, this flexibility significantly impacts project cost and completion time. The ability of clients to modify requirements at any stage introduces unpredictability, making traditional cost estimation techniques ineffective in agile environments. Since the product owner controls these changes, project teams struggle with accurately forecasting development efforts.

Existing cost estimation models, such as COCOMO and Function Point Analysis, primarily focus on overall team performance and fail to address individual developer contributions. These models lack precision in tracking developer-specific efforts, leading to inaccurate projections of cost and time. Moreover, they do not account for variations in developer expertise, learning curves, and efficiency, which are crucial factors in agile project execution.

To address these limitations, this study introduces HuBiCEM, a cost estimation model that evaluates individual developer performance instead of relying solely on team-based metrics. By categorizing performance across multiple dimensions and applying Linear Regression, HuBiCEM provides a more detailed and accurate prediction of cost and time in agile projects.

We have implemented HuBiCEM in real-world projects and conducted a survey study with software development teams to assess its effectiveness. The findings demonstrate that HuBiCEM significantly improves cost estimation accuracy compared to traditional methods, making it a valuable tool for agile project management.

III. RELATED WORKS

Cost estimation in agile development has been a significant challenge due to frequent changes in project scope. Various estimation techniques have been developed to address this issue, but none provide a perfect solution.

3.1 COCOMO

The Constructive Cost Model (COCOMO) was introduced by Barry Boehm in 1981. It estimates software effort based on the size of the project measured in KLOC (thousands of lines of code). The model has three modes:
Organic Mode: For small, simple projects (2-50 KLOC).
Semi-Detached Mode: For medium-sized projects (50-300 KLOC).
Embedded Mode: For large, complex systems (>300 KLOC).
COCOMO provides general estimates but struggles with agile environments where frequent requirement changes impact effort calculations.

3.2 Function Point Analysis (FPA)

FPA measures software size based on functionalities rather than lines of code. It classifies software components, such as Internal Logical Files (ILFs), External Outputs (EOs), and External Inputs (EIs), assigning complexity levels to each. The total function points (FPs) are calculated using a Value Adjustment Factor (VAF). While effective for structured projects, FPA lacks adaptability to agile development, where user stories evolve dynamically

3.3 SS-Model

The SS-Model (Shariq Screen Model) is a structured effort estimation technique designed for agile development. It categorizes project modules as Easy, Average, or Difficult, based on past developer experience with similar tasks. The model incorporates a review session where stakeholders estimate cost and time by analyzing previous work and team expertise. SS-Model improves estimation accuracy and helps manage cost overruns caused by frequent change requests. However, it lacks adaptability for individual developer tracking and does not integrate predictive analytics, limiting its precision in evolving agile environments.

IV. PROPOSED MODEL

We introduce HuBiCEM (Hunar Bin Cost Evaluation Model), a prediction-based cost estimation framework designed specifically for individual developer performance tracking in agile projects. Unlike traditional models that evaluate team-based productivity, HuBiCEM measures each developer's contribution individually based on task category, performance trends, and historical data. It then applies Polynomial Regression to forecast future effort, cost, and time estimates for ongoing and upcoming sprints.

HuBiCEM is designed to integrate into agile development workflows, particularly in projects where developer performance varies significantly across different areas, such as backend, frontend, API integration, AI development, and deployment. By tracking developer-specific performance metrics, the model provides more granular cost estimation and improves resource allocation accuracy.

4.1 HuBiCEM Process

The HuBiCEM model follows a structured process to estimate project cost and time. The key steps include:

Step 1: Historical Data Collection
- The system gathers developer performance data from previous sprints.
- Performance is recorded per developer, task, category, expected time, and actual completion time.
- The dataset includes information about sprint-wise developer efficiency in different task categories such as Frontend, Backend, Database, OpenAI Integration, API Development, UI/UX, and Deployment.

Step 2: Task Categorization
- All tasks within a sprint are classified into categories (e.g., API Development, Frontend, Backend, OpenAI Integration).
- Each story point is assigned to a specific developer based on their area of expertise.
- Categories help in training the model for individual performance estimation.

Step 3: HuBi Table Generation
- The model applies Polynomial Regression on historical performance data to generate a HuBi Table, which includes:
- Developer-wise predictive equations in the form ex + c, where e represents estimated effort, x represents historical sprint performance, and c is an adjustment factor based on past efficiency.
- The HuBi Table acts as a lookup table for estimating the time required for future sprints based on developer-specific efficiency trends.

Step 4: Sprint Prediction & Cost Estimation
- Using the HuBi Table, the system estimates the time and cost required for future sprints.
- The prediction considers task complexity, historical efficiency, and skill-based performance variations.
- The model provides realistic estimates, reducing uncertainties in agile cost planning.

4.2 Implementation

To validate the effectiveness of HuBiCEM, we implemented the model in a real-world AI Assistant project.

Project Overview
- A USA-based client required an AI-powered customer representative for his automotive business, 1016industries.
- The AI Assistant needed to welcome customers, provide product details, visualize parts on cars, and process orders.

- The assistant was integrated into a website, social media chatbots, and Raspberry Pi-based kiosks at physical locations.
- The core development tasks involved frontend development, backend APIs, OpenAI integration, chatbot development, and deployment.

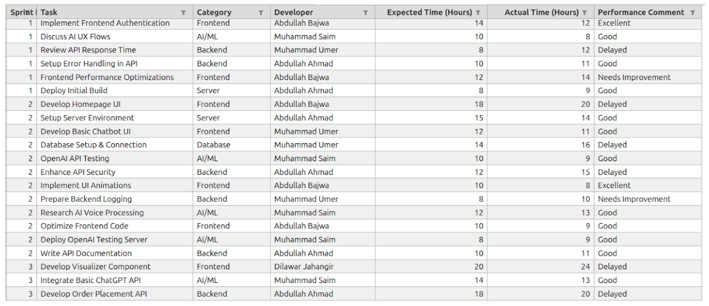
Project Sprints and Development Phases

- The project was planned over 15 sprints (1 year and 3 months).
- Each sprint consisted of story points, assigned to developers based on their expertise.
- Major project tasks included:
    - Developing the homepage and UI (Frontend)
    - Creating a 3D visualizer to display carbon fiber parts (Frontend, API)
    - Developing the backend for order processing and CRM integration (Backend, Database)
    - Building the AI-powered assistant (Python, OpenAI, NLP)
    - Integrating OpenAI's real-time API for voice-based interaction (Python, AI)
    - Embedding the assistant on multiple platforms (Website, Social Media, Kiosk Devices)

4.3 Data Collection for HuBiCEM Training

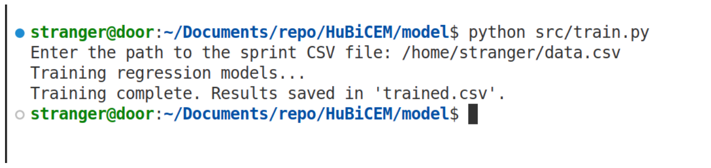
HuBiCEM is trained using historical sprint data, formatted as csv file withthese columns:

- Sprint Number (1 to 15)
- Task Name (e.g., "OpenAI API Integration")
- Category (Frontend, Backend, Database, OpenAI)
- Assigned Developer
- Expected Time (hours)
- Actual Time (hours)
- Performance Feedback



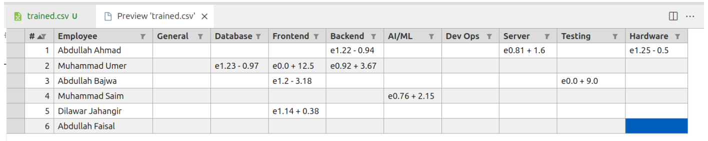
Sprint	Task	Category	Developer	Expected Time (Hours)	Actual Time (Hours)	Performance Comment
1	Implement Frontend Authentication	Frontend	Abdullah Bajwa	14	12	Excellent
1	Discuss AI UX Flows	AI/ML	Muhammad Saim	10	8	Good
1	Review API Response Time	Backend	Muhammad Umer	8	12	Delayed
1	Setup Error Handling in API	Backend	Abdullah Ahmad	10	11	Good
1	Frontend Performance Optimizations	Frontend	Abdullah Bajwa	12	14	Needs Improvement
1	Deploy Initial Build	Server	Abdullah Ahmad	8	9	Good
2	Develop Homepage UI	Frontend	Abdullah Bajwa	18	20	Delayed
2	Setup Server Environment	Server	Abdullah Ahmad	15	14	Good
2	Develop Basic Chatbot UI	Frontend	Muhammad Umer	12	11	Good
2	Database Setup & Connection	Database	Muhammad Umer	14	16	Delayed
2	OpenAI API Testing	AI/ML	Muhammad Saim	10	9	Good
2	Enhance API Security	Backend	Abdullah Ahmad	12	15	Delayed
2	Implement UI Animations	Frontend	Abdullah Bajwa	10	8	Excellent
2	Prepare Backend Logging	Backend	Muhammad Umer	8	10	Needs Improvement
2	Research AI Voice Processing	AI/ML	Muhammad Saim	12	13	Good
2	Optimize Frontend Code	Frontend	Abdullah Bajwa	10	9	Good
2	Deploy OpenAI Testing Server	AI/ML	Muhammad Saim	8	9	Good
2	Write API Documentation	Backend	Abdullah Ahmad	10	11	Good
3	Develop Visualizer Component	Frontend	Dilawar Jahangir	20	24	Delayed
3	Integrate Basic ChatGPT API	AI/ML	Muhammad Saim	14	13	Good
3	Develop Order Placement API	Backend	Abdullah Ahmad	18	20	Delayed

This data is then processed using train.py, which trains the regression model and generates the HuBi Table.



```
● stranger@door:~/Documents/repo/HuBiCEM/model$ python src/train.py
  Enter the path to the sprint CSV file: /home/stranger/data.csv
  Training regression models...
  Training complete. Results saved in 'trained.csv'.
○ stranger@door:~/Documents/repo/HuBiCEM/model$ █
```

Following is a generated output by HuBi model train.py:



trained.csv U   Preview 'trained.csv'

#	Employee	General	Database	Frontend	Backend	AI/ML	Dev Ops	Server	Testing	Hardware
1	Abdullah Ahmad				e1.22 - 0.94			e0.81 + 1.6		e1.25 - 0.5
2	Muhammad Umer		e1.23 - 0.97	e0.0 + 12.5	e0.92 + 3.67					
3	Abdullah Bajwa			e1.2 - 3.18					e0.0 + 9.0	
4	Muhammad Saim					e0.76 + 2.15				
5	Dilawar Jahangir			e1.14 + 0.38						
6	Abdullah Faisal									

The HuBi Table is a structured dataset generated by train.py, containing developer-specific performance prediction equations in the form ex + c, where e represents estimated effort, x represents historical sprint performance, and c is an adjustment factor derived from Linear Regression. train.py processes sprint-wise historical data, categorizes tasks, and applies regression analysis on individual developer performance across different domains such as Frontend, Backend, API Development, and AI Integration. The resulting HuBi Table serves as a reference for predict.py, enabling precise time and cost estimations for future sprints based on developer efficiency trends.

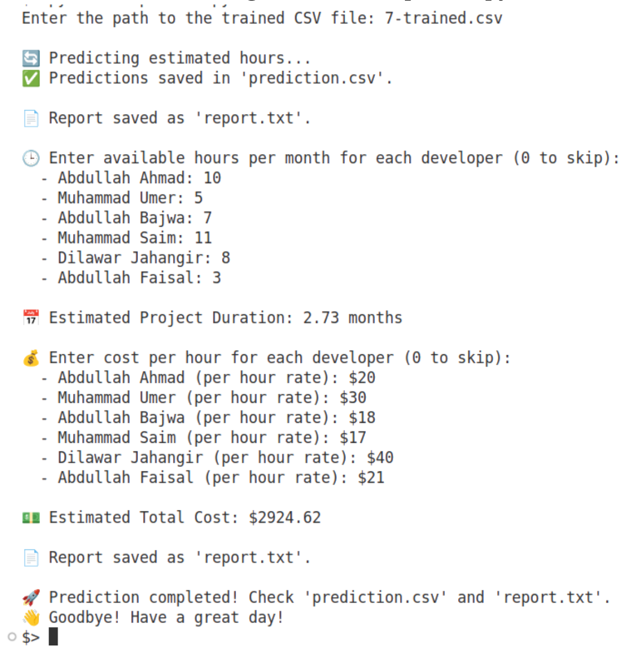
4.3 Predicting future sprints with HuBiCEM

Once the model is trained, it is used for future sprint estimation using predict.py.

Output includes:

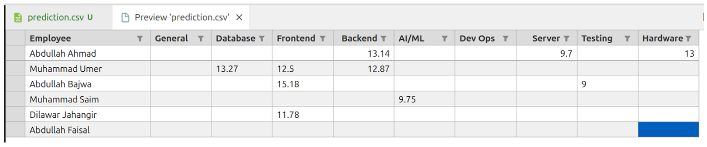
- Predicted Effort per Developer
- Estimated Cost per Sprint
- Total Project Duration Projection
-

Take a look at following execution of predict.py



```
Enter the path to the trained CSV file: 7-trained.csv

🔄 Predicting estimated hours...
✅ Predictions saved in 'prediction.csv'.

📄 Report saved as 'report.txt'.

🕒 Enter available hours per month for each developer (0 to skip):
 - Abdullah Ahmad: 10
 - Muhammad Umer: 5
 - Abdullah Bajwa: 7
 - Muhammad Saim: 11
 - Dilawar Jahangir: 8
 - Abdullah Faisal: 3

📅 Estimated Project Duration: 2.73 months

💰 Enter cost per hour for each developer (0 to skip):
 - Abdullah Ahmad (per hour rate): $20
 - Muhammad Umer (per hour rate): $30
 - Abdullah Bajwa (per hour rate): $18
 - Muhammad Saim (per hour rate): $17
 - Dilawar Jahangir (per hour rate): $40
 - Abdullah Faisal (per hour rate): $21

💵 Estimated Total Cost: $2924.62

📄 Report saved as 'report.txt'.

🚀 Prediction completed! Check 'prediction.csv' and 'report.txt'.
👋 Goodbye! Have a great day!
○ $> █
```

It generates a predicted.csv file also:



prediction.csv U   Preview 'prediction.csv'

Employee	General	Database	Frontend	Backend	AI/ML	Dev Ops	Server	Testing	Hardware
Abdullah Ahmad				13.14			9.7		13
Muhammad Umer		13.27	12.5	12.87					
Abdullah Bajwa			15.18					9	
Muhammad Saim					9.75				
Dilawar Jahangir			11.78						
Abdullah Faisal									

The predict.py script utilizes the HuBi Table generated by train.py to forecast the estimated time, cost, and duration for upcoming sprints. It first prompts the user to input the path to the trained dataset (e.g., 7-trained.csv), then applies Polynomial Regression-based prediction equations for each developer to estimate effort in future tasks.

The script then asks for available work hours per developer per month to calculate the estimated project duration. Additionally, it collects cost per hour for each developer to compute the total estimated project cost. The results are saved in two files:

- prediction.csv – Contains detailed per-task time estimates for each developer.
- report.txt – Summarizes the total estimated project duration and cost.

By leveraging individual performance trends stored in the HuBi Table, predict.py ensures accurate resource allocation and cost forecasting, making it an essential tool for sprint planning in agile development.

The model ensures that sprint planning and resource allocation are accurate and cost-effective.

4.3 Availability & Integration

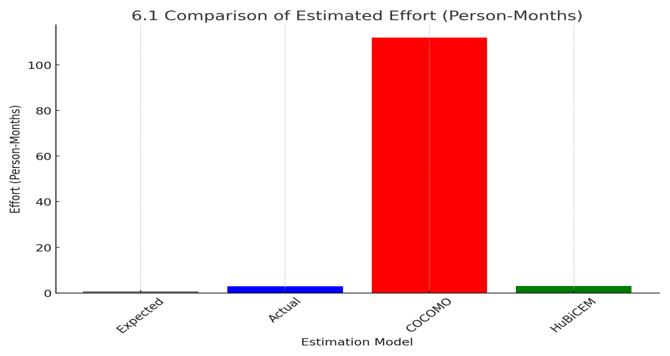
HuBiCEM is implemented as a Python-based tool and can be accessed from:
- GitHub Repository: https://github.com/HuBiCEM/model
- Project Website: https://hubicem.github.io
The complete source code and getting started documentation is available with examples for integration into agile project management systems.

V. RESULTS AND COMPARISON

This section presents a comparative analysis between HuBiCEM and the traditional COCOMO model in terms of effort estimation, time prediction, and cost forecasting. The results have been derived using real-world project data to assess the accuracy of both models.

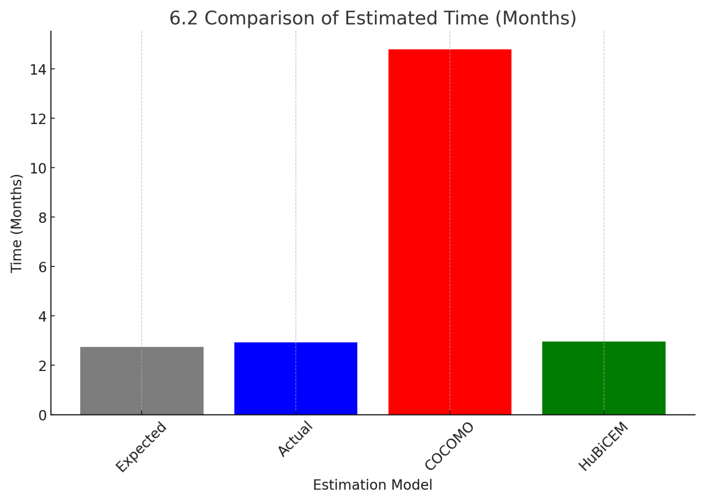
5.1 Effort Estimation Comparison

The effort estimation in terms of person-months varies significantly across models. COCOMO estimates a considerably higher effort, while HuBiCEM is much closer to the actual project effort. The comparison shows that HuBiCEM improves estimation accuracy by tracking individual developer performance rather than relying on generalized team-based estimations.



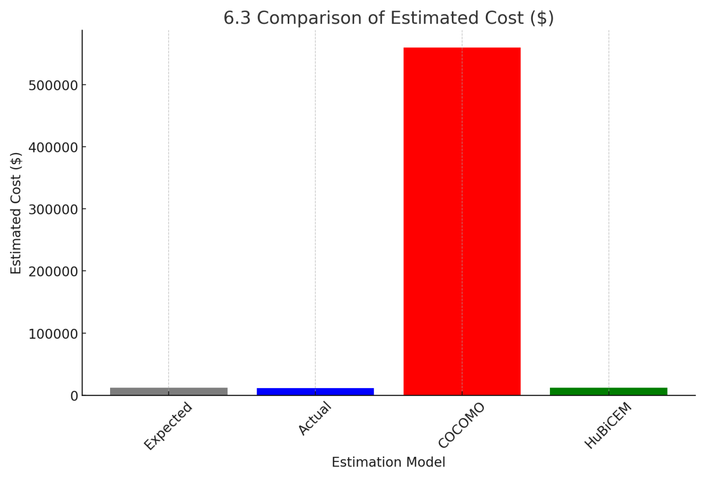
5.2 Time Estimation Comparison

One of the key issues with COCOMO is that it often overestimates project duration, leading to inefficient project planning. The HuBiCEM model predicts a more realistic time estimation, aligning closely with real-world sprint performance.



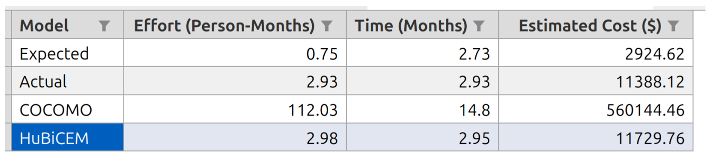
5.3 Cost Estimation Comparison

The financial impact of estimation inaccuracies is significant in software projects. COCOMO predicts much higher costs, which can mislead stakeholders into over-budgeting or misallocating resources. HuBiCEM provides a more precise cost estimate, ensuring that development efforts remain within expected financial constraints.



5.4 CSV Data Table

To facilitate further analysis, the CSV file containing the comparative results is provided, showcasing the values for Expected, Actual, COCOMO, and HuBiCEM estimations.



Model	Effort (Person-Months)	Time (Months)	Estimated Cost ($)
Expected	0.75	2.73	2924.62
Actual	2.93	2.93	11388.12
COCOMO	112.03	14.8	560144.46
HuBiCEM	2.98	2.95	11729.76

5.5 Accuracy Comparison

The results demonstrate that COCOMO consistently overestimates project effort, time, and cost, primarily due to its reliance on source lines of code (SLOC) as the core metric. Agile development requires dynamic cost estimation methods, making HuBiCEM more suitable for modern software projects.

COCOMO is rigid and does not adapt well to Agile projects.
HuBiCEM provides sprint-wise tracking, making it more adaptive and responsive to real-time project dynamics.
COCOMO does not consider developer-specific performance, whereas HuBiCEM applied regression-based prediction for individual contributions.
HuBiCEM reduces uncertainty by aligning estimations with real project data, preventing unnecessary cost and time overruns.
The results confirm that HuBiCEM is a more reliable and effective model for Agile software development.

VI. CONCLUSION

The study presents a comprehensive comparison between HuBiCEM and COCOMO, demonstrating the effectiveness of predictive developer-based estimation models in Agile software development. The findings show that COCOMO overestimates cost and effort, making it less suitable for dynamic Agile environments.

On the other hand, HuBiCEM proves to be significantly more accurate, as it accounts for individual developer performance, task complexity, and sprint-wise tracking. The model enhances estimation accuracy, making it a more efficient and scalable approach for Agile project planning.

The study highlights that traditional estimation models like COCOMO were designed for structured software projects, whereas HuBiCEM provides a modern, adaptable solution tailored for Agile methodologies.

By integrating machine learning predictions, sprint tracking, and real-time performance evaluation, HuBiCEM ensures that effort, time, and cost are estimated with higher precision, leading to improved resource allocation, reduced project risks, and better cost control.

APPENDIX

No additional materials are provided in the appendix for this study..

ACKNOWLEDGMENT

The authors would like to express their gratitude to the software development teams who participated in the validation of HuBiCEM. Their feedback and industry insights were invaluable in improving the accuracy of the model.

AUTHORS

Abdullah Ahmad – Department of Artificial Intelligence, UMT Lahore, Email: alif.abdullah.ahmad@gmail.com

Abdullah Bajwa – Independent Researcher, Email: f2021376127@umt.edu.pk

Corresponding Author: Abdullah Ahmad, Email: alif.abdullah.ahmad@gmail.com

.