International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Cultivating Personalized Learning: A Web-Based Data Dashboard and Analytics Using Python with Streamlit and Pandas

Jennifer P. Pilante[1], Salma Anika[2], Chukwueloka P. Abanobi[3], Christian Lee V. Sam[4], Carmelita H. Benito[5], Jesus S. Paguigan[6]

[1,2,3,4]MIT Student, College of Computing and Information Technology, Adamson University
[5,6]Professor, College of Computing and Information Technology, Adamson University

Abstract
This research aims to develop a system that cultivates personalized learning in data dashboards and analytics using Python with Streamlit and Pandas libraries. The system filled the gaps on what tools educators could use to keep up with the fast-changing education—implementing the educator's methodologies with customized learning from learners' needs.
The researchers enforced the Waterfall Methodology Model. It is aligned since the system was developed in swift and short only. The requirements are minimal, the design is simple and user-friendly, and the implementation, testing, and deployment were swift but necessary. Additionally, the researchers utilized Likart 5-point scale evaluation to test the effectiveness of the system on educators.
The evaluators found the system overall satisfactory with its functionality, usability, reliability, efficiency, and portability. The system is an open-source tool dashboard designed for educators to visualize assessment data, helping to identify learners' needs and strengths for personalized learning. It features CSV export (to reduce runtime and cloud-based errors), achievement rankings, and report printing, enhancing adaptability for both educators and students.
The researchers recommend integrating artificial intelligence (AI) and automation into the system. This research addresses the diverse learning methodologies in education and the need for technology to aid educators. Using Python with Streamlit and Pandas to create a tool for measuring data-driven analytics for personalized learning in schools and universities.

Keywords: Personalized Learning, Python Streamlit and Pandas, Data-driven Analytics and Dashboard, and Web-based system

1. Introduction

The personalized learning framework customizes students' educational experiences by recognizing their strengths and weaknesses individually, and it can be determined using a data dashboard and analytics. A data set through mining, which may be big or small, is irrelevant if unstructured as an analytical dashboard. This set can be collected from school administrators and learning Management Systems (LMS). Nevertheless, how can educators leverage it to maximize the relations and improve the students' learning experience? With a comprehensive data dashboard and analytics employing Python with Streamlit and

Pandas.

There are a variety of strategies for implementing personalized learning, and one of them to constitute is data-driven analytics. It bases students' performance on mined data and compiles them into a comprehensive analytical dashboard. This results in a strong and competent foundation for personalized learning because the data are accurately measurable. There are open sources (automated, such as chatbots and machine learning websites and software, or manual, such as in MS Excel) for educators to utilize the analytics dashboard; this minimizes costs and empowers support from their institution. Though digital literacy may be a gap, educators need to become more familiar with the usefulness of data as learning evidence, and Python with Streamlit and Pandas has made it effortless.

Streamlit and Pandas are game-changers in the field of education, an open-source Python library to create and share custom web apps and deploy powerful data applications for machine learning (ML) and data science. Promoting a minimal and straight-to-the-point line of codes to execute a data analytics output. It supports low latency data-read runtime for it uses CSV file uploading and is flexible for cloud computing integration. In addition, Python is a flexible programming language and library where a user can develop a program, even with four pillars of object-oriented programming: Inheritance, Polymorphism, Encapsulation, and Data Abstraction.

2. Methodology

The researchers applied the Waterfall Methodology Model to the implemented system. The said methodology is a traditional methodology for simple software development. The development process phases are executed linearly and, one by one, completed prior to the previous one (Gapunenko, 2023).

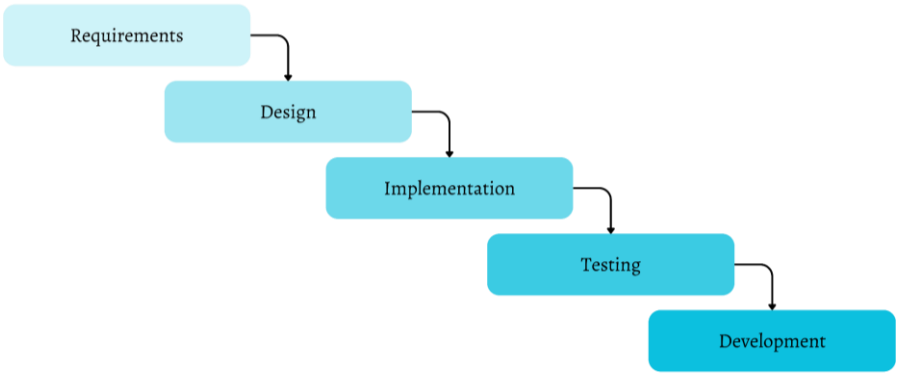

Fig.1 Waterfall Methodology

Requirements

The system, a comprehensive web-based data dashboard and analytics for personalized learning requires a Windows desktop and laptop computer device with Windows 24.2 megabytes storage or macOS 10.9 40.9 megabytes storage. For software, Visual Studio Code, version 1.95, is installed with Python 3.11.3 programming language (Windows 64-bit, recommended, or 32-bit, or macOS 10.9 64-bit), Streamlit, and Pandas libraries. Microsoft Excel or Google Spreadsheet for true-data consolidation. Lastly, since the system is web-based, the internet is a must with low latency.

Design

The research focuses on developing a comprehensive web-based data dashboard and analytics with Python Streamlit to cultivate personalized learning.

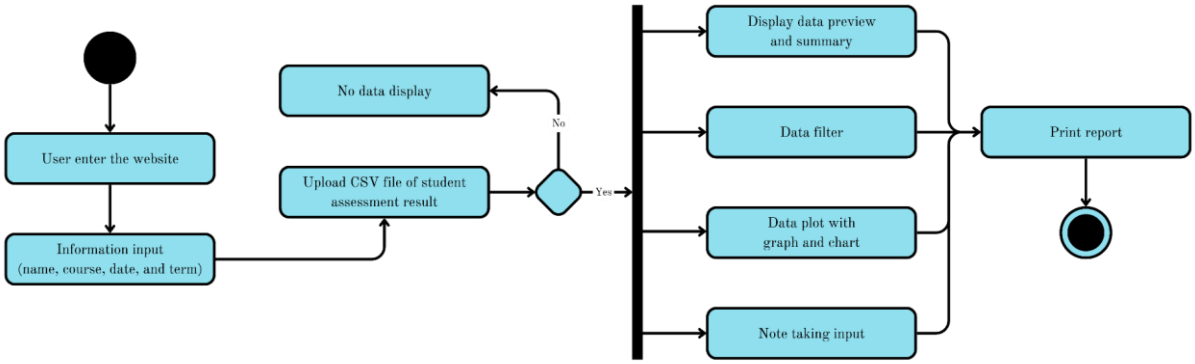

Fig.2 The UML-Activity Diagram of the System

The system design is illustrated using the UML Activity Diagram, as seen in *Figure 2*. The activity or the process is when the user enters the website. They will also input the information needed according to the available fields. CSV upload allows them to visualize the data analytics by integrating the Streamlit library. The information and files asked are for record-keeping, and printing the website enables the educator to save multiple reports.

Implementation

The researchers utilized Python programming language and ran with imported Streamlit, Pandas, and Plotly libraries. The researchers applied the four (4) pillars of object-oriented programming (OOP) as well to stimulate the integrity of the set of codes and values transposed to the variables.

Main class: main.py

```
import streamlit as st
from encapsulation import teacherInfo
from polymorphism import Term1, Term2, Term3, TermSum
from abstract import DataDashboard
from inheritance import derivedNote

st.title("Comprehensive Data Dashboard and Analytics for Personalized Learning")

class main(): #main class
    def __init__(user):
        #get encapsulated variables from encapsulation.py
        user.info = teacherInfo()
        colName, colCourse, colDate = st.columns(3)

        #set private variable of encapsulation.py
        name = user.info.setName
        course = user.info.setCourse
```

```
    date = user.info.setDate
    with colName: name = st.text_input("Name:")
    with colCourse: course = st.text_input("Course:")
    with colDate:
      date = st.date_input("Date:")
      date_str = date.strftime("%Y/%m/%d")  # Format: YYYY-MM-DD
    colTerm, colT1, colT2, colT3, colTS = st.columns(5)

    #Create checkboxes for each term from polymorphism.py
    with colTerm: st.write ("Term:")
    with colT1: term1 = st.checkbox("Term 1")
    with colT2: term2 = st.checkbox("Term 2")
    with colT3: term3 = st.checkbox("Term 3")
    with colTS: termSum = st.checkbox("Summer Term")
    selected_term = []
    if term1: selected_term.append(Term1().this())
    if term2: selected_term.append(Term2().this())
    if term3: selected_term.append(Term3().this())
    if termSum: selected_term.append(TermSum().this())
    term_str = ", ".join(selected_term)

    #Display input value to protected variable from eancapsulation.py and variable from polymorphism.py
    if name and course and date_str and term_str:
      st.subheader(f"Hi, teacher {name}! This is analytics report on {date_str} for {course} - {term_str}.")

#Get and run abstract.py class
class analyticsPage():
  def __init__(DataDasboard):
    dashboard = DataDashboard()
    uploaded_file = st.file_uploader("Choose a CSV file", type="csv")
    dashboard.loadFile(uploaded_file)
    dashboard.filePreview()
    dashboard.fileSummary()
    dashboard.fileFilter()
    dashboard.filePlot()

#Get and run inheritance.py class
class notes():
  def __init__(baseNote):
    st.subheader("Notes:")
```

```
        handler = derivedNote()
        user_input = st.text_area("Enter your important notes here:", height=100)
        if user_input:
            handler.update_input(user_input)
        st.write(handler.display_input())
        st.subheader("Press CTRL + 'P' to print the report.")

#Run the system
if __name__ == "__main__":
    main()
    analyticsPage()
    notes()
```

Encapsulation: encapsulation.py

```
#private variable set for encapsulation
class teacherInfo:
    def __init__(user):
        #Initialize private variables for text input
        user._uName = None
        user._uCourse = None
        user._uDate = None
    #Set private variable from text input
    def setName(user, name): user._uName = name
    def setCourse(user, course): user._uCourse = course
    def setDate(user, date): user._uDate = date
    #Get and return private variable from text input
    def getName(user): return user._uName()
    def getCourse(user): return user._uCourse()
    def getDate(user): return user._uDate()
```

Polymorphism: polymorphism.py

```
#Multiple variables for one value from text input, polymorphism
class Term:
    def this(terms): return "Terms"
class Term1(Term):
    def this(terms): return "Term 1"
class Term2(Term):
    def this(terms): return "Term 2"
class Term3(Term):
```

```
    def this(terms): return "Term 3"
class TermSum(Term):
    def this(terms): return "Term Summer"
```

Inheritance: inheritance.py

```
#Inherits the method, variable, and value from one class to another (main)
class baseNote:
    def __init__(user):
        user._user_input = ""
    def update_input(user, input_value):
        user._user_input = input_value
    def get_input(user):
        return user._user_input
#Derived class
class derivedNote(baseNote):
    def display_input(user):
        return f"{user.get_input()}"
```

Abstract: abstract.py

```
import streamlit as st
import pandas as pd
import plotly.express as px
from abc import ABC, abstractmethod

class AbstractDataDashboard(ABC):
    @abstractmethod
    def loadFile(self, uploaded_file): pass
    @abstractmethod
    def filePreview(self): pass
    @abstractmethod
    def fileSummary(self): pass
    @abstractmethod
    def fileFilter(self): pass
    @abstractmethod
    def filePlot(self, filtered_df): pass

class DataDashboard(AbstractDataDashboard):
    def __init__(self): self.df = None
    def loadFile(self, uploaded_file):
```

```
        if uploaded_file is not None:
            self.df = pd.read_csv(uploaded_file)
    def filePreview(self):
        if self.df is not None:
            st.subheader("Data Preview")
            st.write(self.df)
            lowScore = self.df['Score'].max()
            lowStudent = self.df.loc[self.df['Score'] == lowScore, 'name'].iloc[0]
            st.success(f"The highest Score is {lowScore}, Student: {lowStudent}.")
            lowScore = self.df['Score'].min()
            lowStudent = self.df.loc[self.df['Score'] == lowScore, 'name'].iloc[0]
            st.error(f"The lowest Score is {lowScore}, Student: {lowStudent}.")
    def fileSummary(self):
        if self.df is not None:
            st.subheader("Data Summary")
            dfDesc = self.df.describe().round(2)
            st.write("This table is the overall summarization of the data, including Standard Deviation and
Mean.")
            st.write(dfDesc)
    def fileFilter(self):
        if self.df is not None:
            st.subheader("Filter Data")
            selected_column = st.selectbox("Select column to filter by", self.df.columns)
            selected_value = st.selectbox("Select value", self.df[selected_column].unique())
            filtered_df = self.df[self.df[selected_column] == selected_value]
            st.write(filtered_df)
            return filtered_df
        return None
    def filePlot(self):
        if self.df is not None:
            st.subheader("Plot Data")
            x_column = st.selectbox("Select x-axis column", self.df.columns)
            y_column = st.selectbox("Select y-axis column", self.df.columns)
            if st.button("Generate Plot"):
                st.line_chart(self.df.set_index(x_column)[y_column])
                if 'name' in self.df.columns and 'Score' in self.df.columns:
                    fig = px.pie(self.df, names='name', values='Score')
                    st.plotly_chart(fig)
```

Testing

Ensuring the uploaded file type is limited to CSV and only one file is uploaded at a time. It is fast and efficient with data parsing, facilitating fast data reading and writing. CSV format is compact, taking up

less space than other file formats, making them useful for storing and sharing large amounts of data. Hence, the data processing is in low latency to reduce runtime errors. The system also ensures that data displays true output from true data by visualizing the imported file. The tables, dashboards, analytics, and charts were tested to verify their usability and efficiency from the data imported.

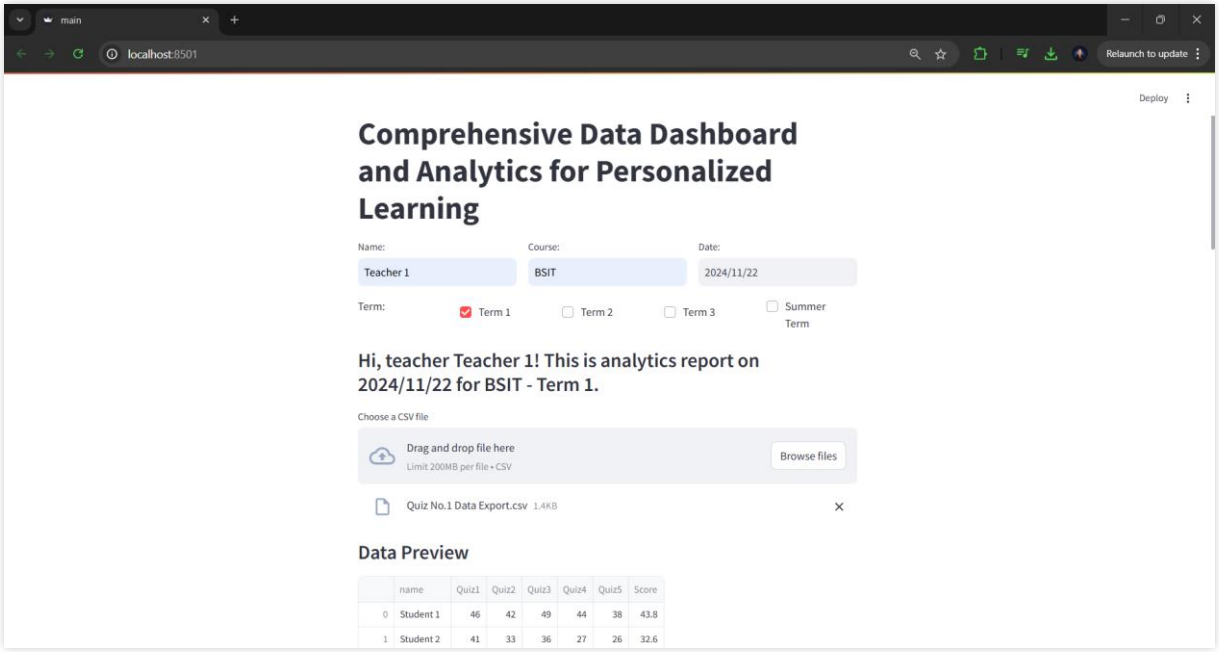

Fig.3 The educator input details for report-keeping

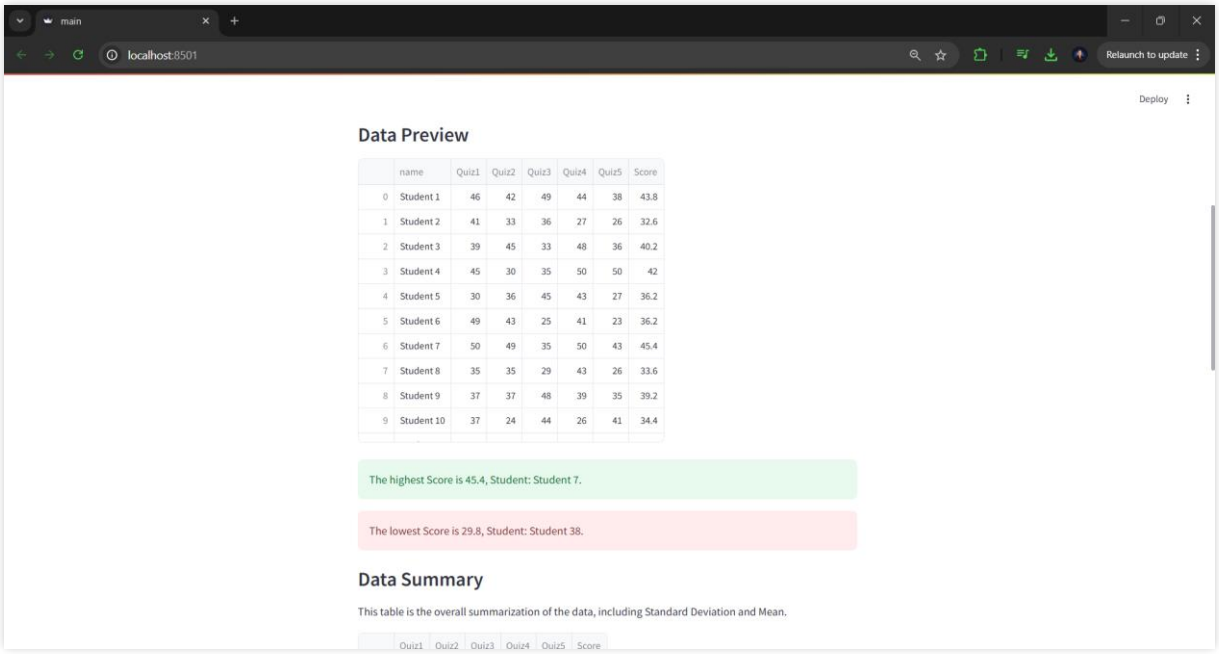

Fig.4 Data table preview with the identification of student ranking (highest and lowest)

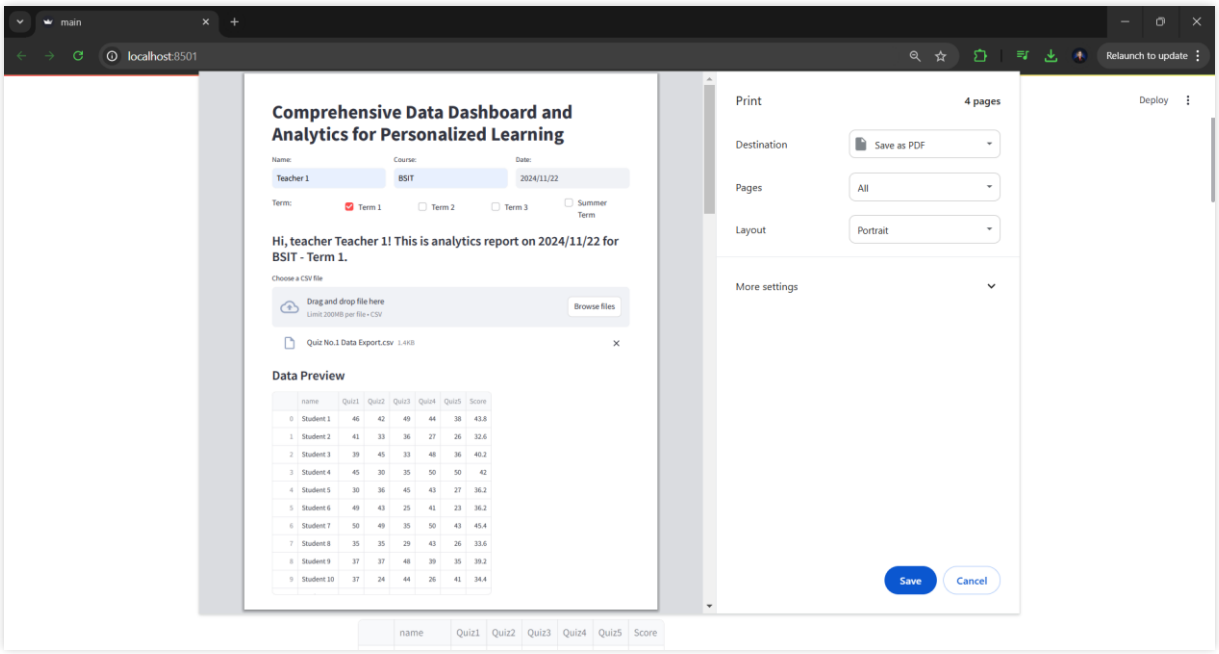

Fig.5 Printing of data dashboard and analytics for future use by educators

Deployment

The beneficiaries of Cultivating Personalized Learning: A Comprehensive Web-based Data Dashboard and Analytics with Python Streamlit are educators. Upon conducting the testing phase, the system was evaluated randomly by five (5) educators. These educators are from university offices focusing on learning freedom using technology; applying data-driven analytics is also part of pedagogy.

Table 1. The system evaluation using the Likart 5-point Scale

Characteristics	Mean	Verbal Interpretation
Functionality The system serves the demand and needs of users.	4.4	Satisfactory
Usability The system achieved the defined goal effectively and efficiently.	4.2	Satisfactory
Reliability The system is performing consistently well.	3.8	Neutral
Efficiency The system is competent, productive, and well-organized.	4	Satisfactory
Portability The system is accessible and transferable.	4.2	Satisfactory
Overall Weighted Mean	4.12	Satisfactory

Table 1 indicates the evaluation result conducted by the researchers from educators with an overall weighted mean of 4.12, satisfactory.

3. Results and Discussion

The objective of the system is to develop a data-driven analytics dashboard that shows educators their students learning progress to cultivate personalized. This includes a graph, chart, and table that summarizes the assessment of the learners and who is the least and most active among them. The system was developed using Python with Streamlit and Pandas library.

As a result from Table 1, The system evaluation using the Likart 5-point Scale: the system functionality that serves the demands and needs of users is 4.4 (Satisfactory); it achieved the defined goal effectively and efficiently, usability is 4.2 (Satisfactory); it is reliable, 3.8 (Neutral); the system's efficiency is 4, competent, productive and well-organized (Satisfactory); and lastly, portability is 4.2, the system is accessible and transferable (Satisfactory). The overall weighted mean is 4.12, which is satisfactory to educators for data-driven analytics and dashboards to cultivate personalized learning.

4. Conclusions and Recommendations

The Cultivating Personalized Learning: A Web-based Data Dashboard and Analytics Using Python with Streamlit and Pandas provides an open-source tool for educators to use. It allows them to import an assessment result one at a time to visualize the data with tables, graphs, and charts. It empowers them to cultivate personalized learning by identifying the needs, strengths, and weaknesses of learners for a customizable educational experience.

The main features are exporting assessment results using CSV files, visualizing data with tables, charts, and graphs, displaying the least and most achievers of the class (to monitor the learning needs), and printing the report for future usability and reference. Leverage the resources available to maximize the adaptability of learning freedom, not just for educators but for students/learners as well.

The practical implication for this research and the system is the usability of inevitable applications of educational evolution. From what we have known as active and passive learning, education now has a variety of learning and teaching methodologies. But with these fast-paced changes in education, how can technology help educators deliver their methods efficiently?

This research and system nourish the gap on what tool to use to measure data-driven analytics for personalized learning. The implementation of the waterfall methodology model was swift for educators. The programming language executed, Python with Streamlit and Pandas libraries, supported the goal and objective of developing the tool for the realistic essentiality of schools and universities.

The vast influence and trend of artificial intelligence (AI) has changed the perspective of 21st-century technology—the automation it provides to deliver efficient and convenient help to the end-users. The researchers' recommendation is to integrate AI into the system. This will automate the workflow to maximize the dashboard analytics feature, including automation such as chatbots and detecting true and incompatible data. Additionally, integrating AI and automation will guide and help educators apply the trust that AI is reliable to use, mainly regarding data. The educators may be familiarized and confident with the system as a routine.

5. References

1. Kapadia, V. (2024, August 6). How can Big Data Analytics help in delivering personalized learning? GyrusAim LMS. https://www.gyrus.com/blogs/how-can-big-data-analytics-help-in-delivering-personalized-learning
2. GeeksforGeeks. (2024, July 9). Four main object-oriented programming concepts of Java.

GeeksforGeeks. https://www.geeksforgeeks.org/four-main-object-oriented-programming-concepts-of-java

3. Kstudylearning (2024). The role of data analytics in personalized learning is to trailer education for each student. https://kstudylearning.com/the-role-of-data-analytics-in-personalized-learning-tailoring-education-for-each-student
4. Gapunenko, J. (2023, December 28). 5 Most effective Web development methodologies. ADCI Solutions. https://www.adcisolutions.com/knowledge/web-development-methodologies-and-approaches
5. Teachflow. (2023, June 6). The role of Data Analytics in personalized education. https://teachflow.ai/the-role-of-data-analytics-in-personalized-education
6. Magomadov, V. S. (2020). The application of artificial intelligence and Big Data analytics in personalized learning. Journal of Physics: Conference Series, 1691(1), 012169. https://doi.org/10.1088/1742-6596/1691/1/012169
7. Lin, R. (2023 , Jul 11).Becoming a Self-Taught Data Analyst: A DIY Journey into Data Analysis. https://medium.com/pathway-to-data-analysis/becoming-a-self-taught-data-analyst-a-diy-journey-into-data-analysis-c4efececea10
8. Gupta, S. (2023, July 10). 7-Step Guide on How To Learn Data Analysis (as a Beginner). https://www.springboard.com/blog/data-analytics/learn-data-analysis

Licensed under Creative Commons Attribution-ShareAlike 4.0 International License