|  |
| --- |
| **UNIVERSITY MANAGEMENT SYSTEM**    ***A Python Project Submitted***      **In**    **CSE (DATA SCIENCE)**      **by**    **ABHINEET KUMAR JHA**            **Under the Supervision of**  **Vikash Tripathi**  **Asst. Prof., AI**  **Roshni Afshan**  **Asst. Prof., AI-ML**              **NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY,**  **GREATER NOIDA**  **(An Autonomous Institute)**  **Affiliated to**  **DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW**  **June, 2025** |

***Overview of the University***

***Management System Code***

➢ **Overview:**

The code is designed to create a comprehensive University Management System that facilitates the administration of student and faculty records, financial

management, and communication within the university. It provides a userfriendly interface for administrators to perform various tasks efficiently.

Framework and Libraries Tkinter: The primary library used for building the

GUI. It allows for the creation of windows, buttons, labels, and other interactive elements. Regular Expressions (re): Utilized for validating email formats, ensuring that user inputs conform to expected patterns. Key Components Data Structures:

A list that holds dictionaries, each representing a student’s information, such as name, email, phone number, and enrollment details. faculty\_data: Similar to students\_data, this list contains dictionaries for faculty members, storing their personal and professional details. recent\_emails:

is\_valid\_name(name): Checks if the name contains only alphabetic characters and spaces. is\_valid\_email(email): Validates the email format using a regular expression. is\_valid\_phone(phone): Ensures the phone number consists of

exactly 10 digits. is\_valid\_number(value): Checks if the input is a digit, used for fields like student ID and enrollment year. User Authentication:

Login Window Class: Manages the login process for users. It verifies credentials for Admin and Faculty users against predefined values. Upon

successful login, it opens the corresponding dashboard. Admin Dashboard:

Admin Dashboard Class: Provides a central interface for administrators to access various management functionalities. It includes buttons for: Student

Management Faculty Management Finance Management Communication & Portal Access Logout Management Functionalities:

Communication Porta window Class: Facilitates sending emails to all students or faculty members. It includes: Input fields for the sender's name, recipient branch, and email content. Functions to filter recipients based on specified branches. A record-keeping system for recent emails sent.

* **Conclusion:**

This University Management System code is a robust solution for managing various administrative tasks within a university setting. It

combines data management, user authentication, and communication functionalities into a single application, making it easier for

administrators to handle their responsibilities efficiently. The use of a graphical interface ensures that users can navigate the system with

ease, while the underlying data structures and validation mechanisms maintain data integrity and accuracy. Overall, this code serves as a foundational tool for educational institutions looking to streamline their administrative processes.

* **REFERENCES:**

* Python Official Documentation: <https://docs.python.org/3/>

* W3Schools Python: <https://www.w3schools.com/python/>

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to all these individuals for mentoring and supporting me in completing this project University Management System

My teachers Vikash Tripathi Sir and Roshni Afshan Mam for providing me with invaluable insights and direction.

To my parents, their constant encouragement, patience, and understanding have been the pillars of my success.

I am grateful to my friends who contributed ideas and perspectives that enriched the project.

Thank you everyone for shaping this project and enhancing my learning experience.

# FRONT PAGE

class BHUUniversityApp:

def \_\_init\_\_(self, root):

self.root = root self.root.title("Welcome to BHU University") self.root.geometry("450x300") self.root.configure(bg="lightcyan") self.create\_main\_interface()

def create\_main\_interface(self):

tk.Label(self.root, text="Welcome to BHU University", font=("Helvetica", 18, "bold"), fg="navy", bg="lightcyan").pack(pady=30)

tk.Button(self.root, text="Sign in as Admin", bg="darkgreen", fg="white", font=("Arial", 13), width=25, command=lambda: self.open\_login("Admin")).pack(pady=10)

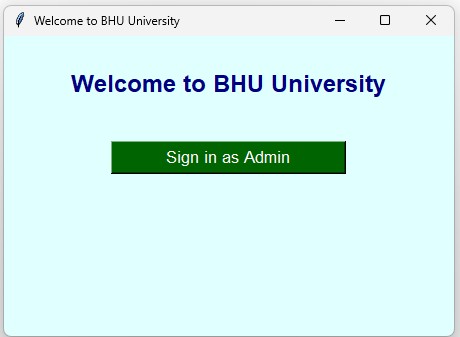
def open\_login(self, user\_type):

login\_root = tk.Toplevel(self.root)

LoginWindow(login\_root,user\_type, open\_dashboard\_callback=self.open\_admin\_dashboard)

def open\_admin\_dashboard(self):

AdminDashboard()



# LOGIN PAGE

class LoginWindow: def \_\_init\_\_(self, master, user\_type, open\_dashboard\_callback=None):

self.master = master self.master.title(f"{user\_type} Login - BHU University") self.master.geometry("350x250") self.master.configure(bg="lightblue") self.user\_type = user\_type self.open\_dashboard\_callback = open\_dashboard\_callback

self.valid\_credentials = {

'Faculty': {'email': 'faculty', 'password': '12345'},

'Admin': {'email': 'admin', 'password': '12345'}

}

self.create\_widgets()

def create\_widgets(self):

tk.Label(self.master, text=f"{self.user\_type} Login", font=("Arial", 16, "bold"), bg="lightblue", fg="navy").pack(pady=10) tk.Label(self.master, text="Email:", font=("Arial", 12), bg="lightblue").pack() self.email\_entry = tk.Entry(self.master, font=("Arial", 12), width=25) self.email\_entry.pack()

tk.Label(self.master, text="Password:", font=("Arial", 12), bg="lightblue").pack() self.password\_entry = tk.Entry(self.master, show="\*", font=("Arial", 12), width=25) self.password\_entry.pack()

tk.Button(self.master, text="Login", bg="navy", fg="white", font=("Arial", 12), width=15, command=self.validate\_login).pack(pady=15)

def validate\_login(self):

email = self.email\_entry.get() password = self.password\_entry.get() correct = self.valid\_credentials[self.user\_type]

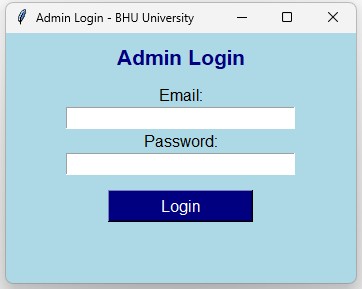
if email == correct['email'] and password == correct['password']:

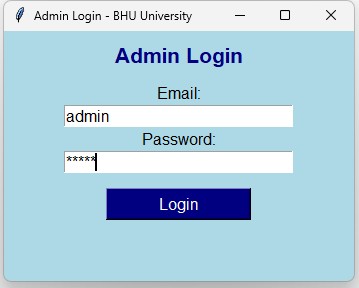
messagebox.showinfo("Login Successful", f"Welcome, {self.user\_type}!") self.master.destroy() if self.user\_type == 'Admin' and self.open\_dashboard\_callback:

self.open\_dashboard\_callback()

else:

messagebox.showerror("Login Failed", "Invalid email or password.")





## ADMIN DASHBOARD

class AdminDashboard:

def \_\_init\_\_(self):

self.root = tk.Toplevel() self.root.title("Admin Dashboard - BHU University") self.root.geometry("450x400") self.root.configure(bg="honeydew")

tk.Label(self.root, text="Admin Dashboard", font=("Helvetica", 18, "bold"), fg="darkgreen", bg="honeydew").pack(pady=20)

self.create\_buttons()

def create\_buttons(self):

button\_style = {"font": ("Arial", 13), "width": 30, "height": 2, "bd": 0}

tk.Button(self.root, text="Student Management", bg="dodgerblue", fg="white", command=self.student\_mgmt, \*\*button\_style).pack(pady=8) tk.Button(self.root, text="Faculty Management", bg="limegreen", fg="white", command=self.faculty\_mgmt, \*\*button\_style).pack(pady=8) tk.Button(self.root, text="Finance Management", bg="orange", fg="white", command=self.open\_finance\_management, \*\*button\_style).pack(pady=8) tk.Button(self.root, text="Communication & Portal Access", bg="mediumorchid", fg="white", command=self.open\_communication\_portal, \*\*button\_style).pack(pady=8) tk.Button(self.root, text="Logout", bg="red", fg="white", command=self.root.destroy, \*\*button\_style).pack(pady=8)

def student\_mgmt(self):

StudentManagementWindow(self.root)

def faculty\_mgmt(self):

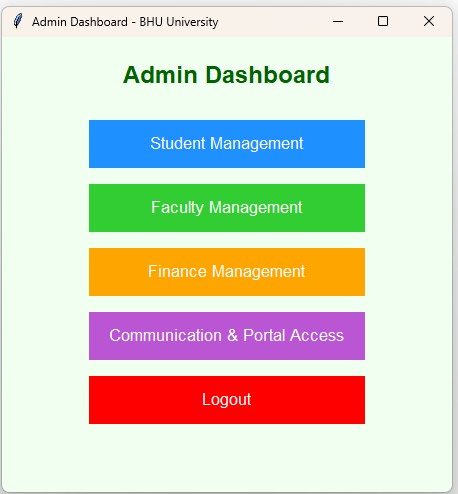
FacultyManagementWindow(self.root)

def open\_finance\_management(self):

FinanceManagementWindow(self.root)

def open\_communication\_portal(self):

CommunicationPortalWindow(self.root)



**Student Management Window Class (Main Interface)**

class StudentManagementWindow:

def \_\_init\_\_(self, parent):

self.root = tk.Toplevel(parent) self.root.title("Student Management - BHU University") self.root.geometry("400x400") self.root.configure(bg="aliceblue")

tk.Label(self.root, text="Student Management", font=("Arial", 16, "bold"), bg="aliceblue", fg="navy").pack(pady=20)

self.add\_buttons() def add\_buttons(self):

buttons = [

("Add Student", self.open\_add\_student, "dodgerblue"),

("Show All Students", self.show\_all\_students, "limegreen"),

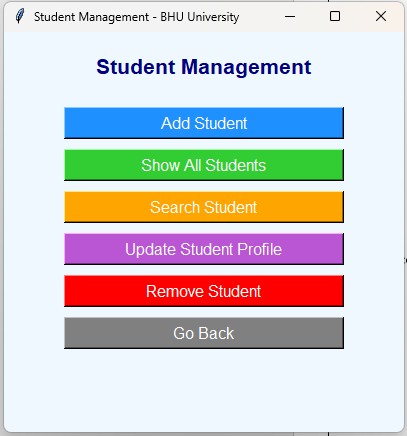
("Search Student", self.search\_student, "orange"),

("Update Student Profile", self.update\_student, "mediumorchid"),

("Remove Student", self.remove\_student, "red"),

("Go Back", self.root.destroy, "gray") ] for text, cmd, color in buttons:

tk.Button(self.root, text=text, bg=color, fg="white", font=("Arial", 12), width=30, command=cmd).pack(pady=5)



**Add Student Functionality (Add Student Window Class)**

class AddStudentWindow:

def \_\_init\_\_(self):

self.root = tk.Toplevel() self.root.title("Add Student - BHU University") self.root.geometry("500x600") self.root.configure(bg="whitesmoke")

self.entries = {}

self.message\_label = tk.Label(self.root, text="", fg="red", font=("Arial", 10, "bold"), bg="whitesmoke") self.message\_label.pack(pady=5)

self.build\_form()

def build\_form(self):

fields = {

"Full Name": is\_valid\_name,

"Date of Birth (DD-MM-YYYY)": None,

"Gender": None,

"Email Address": is\_valid\_email,

"Phone Number": is\_valid\_phone,

"Address": None,

"Student ID / Roll Number": is\_valid\_number,

"Enrollment Year": is\_valid\_number,

## Search Student

def search\_student(self): def find():

query = entry.get().strip().lower() for student in students\_data:

if query in student.get("Full Name", "").lower() or query == student.get("Student ID / Roll Number"):

self.view\_student\_detail(student)

return

messagebox.showerror("Not Found", "No student found with given name or roll number.")

win = tk.Toplevel(self.root) win.title("Search Student") tk.Label(win, text="Enter Name or Roll Number:").pack(pady=5)

## Update Student

def update\_student(self): def search():

query = entry.get().strip().lower() for student in students\_data:

if query == student.get("Student ID / Roll Number") or query in student.get("Full Name", "").lower(): update\_form(student)

return

messagebox.showerror("Not Found", "Student not found.")

def update\_form(student): win = tk.Toplevel(self.root) win.title("Update Profile") entries = {}

uneditable = ["Full Name", "Student ID / Roll Number", "Gender", "Date of Birth (DD-MMYYYY)", "Enrollment Year"]

for key, value in student.items(): if key.startswith("\_"):

continue if key in uneditable:

tk.Label(win, text=f"{key}: {value}").pack(pady=2)

else:

tk.Label(win, text=key).pack() e = tk.Entry(win)

e.insert(0, value)

e.pack() entries[key] = e

def save(): for key, entry in entries.items(): student[key] = entry.get().strip()

## Delete Student

def update\_student(self): def search():

query = entry.get().strip().lower() for student in students\_data:

if query == student.get("Student ID / Roll Number") or query in student.get("Full Name", "").lower(): update\_form(student)

return

messagebox.showerror("Not Found", "Student not found.")

def update\_form(student): win = tk.Toplevel(self.root) win.title("Update Profile") entries = {}

uneditable = ["Full Name", "Student ID / Roll Number", "Gender", "Date of Birth (DD-MM- YYYY)", "Enrollment Year"]

for key, value in student.items(): if key.startswith("\_"):

continue if key in uneditable:

tk.Label(win, text=f"{key}: {value}").pack(pady=2)

else:

tk.Label(win, text=key).pack() e = tk.Entry(win)

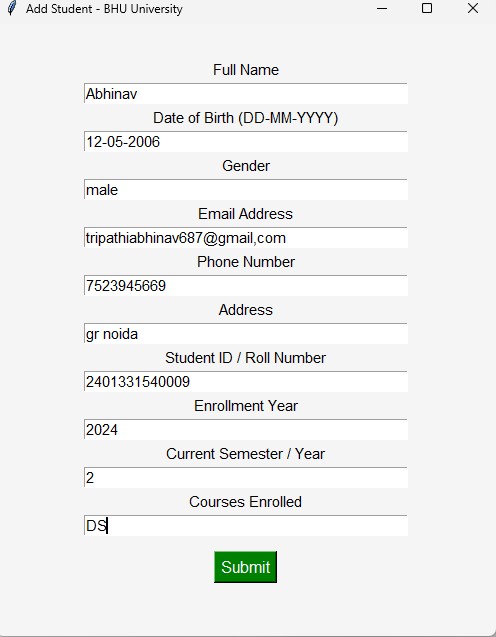
e.insert(0, value)

e.pack() entries[key] = e

def save(): for key, entry in entries.items(): student[key] = entry.get().strip() messagebox.showinfo("Updated", "Student profile updated successfully.") win.destroy()

tk.Button(win, text="Save Changes", command=save).pack(pady=10)

win = tk.Toplevel(self.root) win.title("Search for Update") tk.Label(win, text="Enter Name or Roll Number:").pack(pady=5) entry = tk.Entry(win) entry.pack(pady=5) tk.Button(win, text="Search", command=search).pack(pady=5)



**Faculty Management Window Class (Main Interface)**

**class FacultyManagementWindow:**

**def \_\_init\_\_(self, parent):**

**self.root = tk.Toplevel(parent) self.root.title("Faculty Management - BHU University") self.root.geometry("400x400") self.root.configure(bg="aliceblue")**

**tk.Label(self.root, text="Faculty Management", font=("Arial", 16, "bold"), bg="aliceblue", fg="navy").pack(pady=20)**

**self.add\_buttons()**

**def add\_buttons(self):**

**buttons = [**

**("Add Faculty", self.open\_add\_faculty, "dodgerblue"),**

**("Show All Faculty", self.show\_all\_faculty, "limegreen"),**

**("Search Faculty", self.search\_faculty, "orange"),**

**("Update Faculty Profile", self.update\_faculty, "mediumorchid"),**

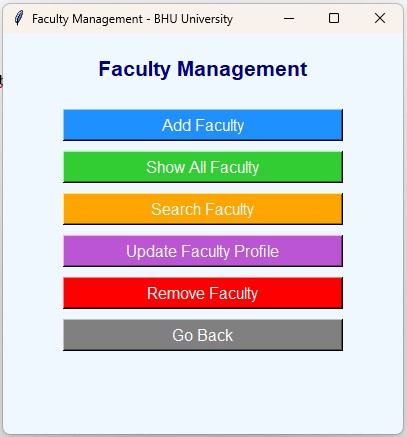
**("Remove Faculty", self.remove\_faculty, "red"),**

**("Go Back", self.root.destroy, "gray")**

**]**

**for text, cmd, color in buttons:**

**tk.Button(self.root, text=text, bg=color, fg="white", font=("Arial", 12), width=30, command=cmd).pack(pady=5)**



### Add Faculty

class AddFacultyWindow:

def \_\_init\_\_(self):

self.root = tk.Toplevel() self.root.title("Add Faculty - BHU University") self.root.geometry("600x700") self.root.configure(bg="whitesmoke") self.entries = {} self.message\_label = tk.Label(self.root, text="", fg="red", font=("Arial", 10, "bold"), bg="whitesmoke") self.message\_label.pack(pady=5) self.build\_form() def build\_form(self):

fields = {

"Full Name": is\_valid\_name,

"Date of Birth": None,

"Gender": None,

"Email Address (Official)": is\_valid\_email,

"Email Address (Personal)": is\_valid\_email,

"Phone Number": is\_valid\_phone

"Employee ID": is\_valid\_number,

"Designation": None,

"Department": None,

"Date of Joining": None,

"Type of Employment": None,

"Subjects Taught": None,

"Years of Experience": is\_valid\_number,

"Highest Degree Achieved": None

} for field, validator in fields.items():

tk.Label(self.root, text=field, font=("Arial", 11), bg="whitesmoke").pack(pady=2) entry = tk.Entry(self.root, font=("Arial", 11), width=45) entry.pack() self.entries[field] = (entry, validator) tk.Button(self.root, text="Submit", bg="green", fg="white", font=("Arial", 12), command=self.validate\_and\_submit).pack(pady=15) def validate\_and\_submit(self):

faculty = {} for field, (entry, validator) in self.entries.items():

value = entry.get().strip() if not value:

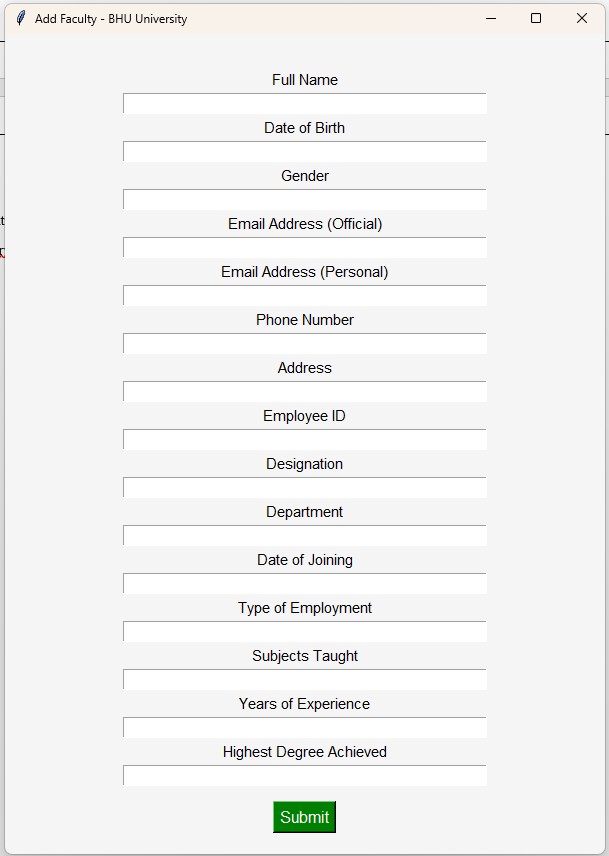
self.message\_label.config(text="Warning: All fields are mandatory!", fg="red")

return if validator and not validator(value):

self.message\_label.config(text=f"Warning: Invalid input in '{field}'", fg="red")

return faculty[field] = value faculty\_data.append(faculty) self.message\_label.config(text="Faculty added successfully!", fg="green") for entry, \_ in self.entries.values():

entry.delete(0, tk.END)



### View All Faculty

def show\_all\_faculty(self): win = tk.Toplevel(self.root) win.title("All Faculty") win.geometry("350x400") for faculty in faculty\_data:

name = faculty.get("Full Name") emp\_id = faculty.get("Employee ID") btn = tk.Button(win, text=f"{name} - {emp\_id}", width=40, command=lambda f=faculty: self.view\_faculty\_detail(f)) btn.pack(pady=3) def view\_faculty\_detail(self, faculty):

win = tk.Toplevel(self.root) win.title("Faculty Detail") win.geometry("400x400") for key, value in faculty.items(): if key.startswith("\_"):

continue tk.Label(win, text=f"{key}: {value}", anchor="w").pack(fill="x", padx=10, pady=2)

### Search Faculty

def search\_faculty(self):

def find():

query = entry.get().strip().lower() for faculty in faculty\_data:

if query in faculty.get("Full Name", "").lower() or query == faculty.get("Employee ID"):

self.view\_faculty\_detail(faculty) return

messagebox.showerror("Not Found", "No faculty found with given name or employee ID.") win = tk.Toplevel(self.root) win.title("Search Faculty") tk.Label(win, text="Enter Name or Employee ID:").pack(pady=5)

entry = tk.Entry(win) entry.pack(pady=5) tk.Button(win, text="Search", command=find).pack(pady=5)

### Update Faculty

def update\_faculty(self): def search():

query = entry.get().strip().lower() for faculty in faculty\_data: if query == faculty.get("Employee ID") or query in faculty.get("Full Name", "").lower():

update\_form(faculty) return

messagebox.showerror("Not Found", "Faculty not found.") def update\_form(faculty): win = tk.Toplevel(self.root) win.title("Update Profile") entries = {} uneditable = ["Full Name", "Employee ID", "Gender", "Date of Birth"] for key, value in faculty.items():

if key.startswith("\_"):

continue if key in uneditable:

tk.Label(win, text=f"{key}: {value}").pack(pady=2)

else:

tk.Label(win, text=key).pack() e = tk.Entry(win)

e.insert(0, value)

e.pack() entries[key] = e def save():

for key, entry in entries.items(): faculty[key] = entry.get().strip() messagebox.showinfo("Updated", "Faculty profile updated successfully.") win.destroy()

tk.Button(win, text="Save Changes", command=save).pack(pady=10) win = tk.Toplevel(self.root) win.title("Search for Update") tk.Label(win, text="Enter Name or Employee ID:").pack(pady=5) entry = tk.Entry(win) entry.pack(pady=5) tk.Button(win, text="Search", command=search).pack(pady=5)

### Delete Faculty

def remove\_faculty(self):

def confirm\_removal(faculty): win = tk.Toplevel(self.root) win.title("Confirm Removal") for key, value in faculty.items(): if key.startswith("\_"):

continue tk.Label(win, text=f"{key}: {value}").pack(pady=2) def remove():

faculty\_data.remove(faculty) messagebox.showinfo("Removed", "Faculty removed successfully."

list\_win.destroy()

tk.Button(win, text="Remove This Faculty", fg="white", bg="red", command=remove).pack(pady=10)

list\_win = tk.Toplevel(self.root) list\_win.title("Select Faculty to Remove") for faculty in faculty\_data:

name = faculty.get("Full Name") emp\_id = faculty.get("Employee ID") btn = tk.Button(list\_win, text=f"{name} - {emp\_id}", width=40, command=lambda f=faculty: confirm\_removal(f)) btn.pack(pady=3)

### Finance Management Window Class (Main Interface)

class FinanceManagementWindow:

def \_\_init\_\_(self, parent):

self.root = tk.Toplevel(parent) self.root.title("Finance Management - BHU University") self.root.geometry("400x400") self.root.configure(bg="aliceblue")

tk.Label(self.root, text="Finance Management", font=("Arial", 16, "bold"), bg="aliceblue", fg="navy").pack(pady=20)

self.add\_buttons() def add\_buttons(self):

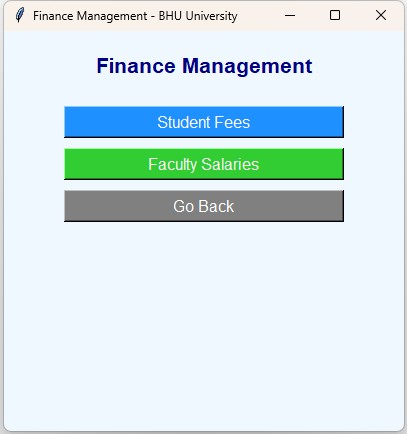
buttons = [

("Student Fees", self.manage\_student\_fees, "dodgerblue"),

("Faculty Salaries", self.manage\_faculty\_salaries, "limegreen"),

("Go Back", self.root.destroy, "gray")] for text, cmd, color in buttons: tk.Button(self.root, text=text, bg=color, fg="white", font=("Arial", 12), width=30,

command=cmd).pack(pady=5)



### Student Fee Management

def manage\_student\_fees(self): win = tk.Toplevel(self.root) win.title("Manage Student Fees") win.geometry("350x400") for student in students\_data:

name = student.get("Full Name") roll = student.get("Student ID / Roll Number") btn = tk.Button(win, text=f"{name} - {roll}", width=40, command=lambda s=student: self.open\_fee\_details(s)) btn.pack(pady=3) def open\_fee\_details(self, student):

win = tk.Toplevel(self.root) win.title("Fee Details") win.geometry("300x230") win.configure(bg="aliceblue")

tk.Label(win, text=f"Fee Details for {student['Full Name']}", font=("Arial", 12, "bold"), bg="aliceblue").pack(pady=10)

paid\_var = tk.BooleanVar() unpaid\_var = tk.BooleanVar()

def on\_paid\_checked(): if paid\_var.get():

unpaid\_var.set(False)

def on\_unpaid\_checked(): if unpaid\_var.get(): paid\_var.set(False)

tk.Checkbutton(win, text="Paid", variable=paid\_var, command=on\_paid\_checked, bg="aliceblue").pack()

tk.Checkbutton(win, text="Unpaid", variable=unpaid\_var, command=on\_unpaid\_checked, bg="aliceblue").pack() tk.Label(win, text="Amount:", bg="aliceblue").pack(pady=5)

amount\_entry = tk.Entry(win) amount\_entry.pack(pady=5) fee\_record = student.get("\_fee\_record", {}) if fee\_record: if fee\_record.get("status") == "Paid":

paid\_var.set(True) elif fee\_record.get("status") == "Unpaid":

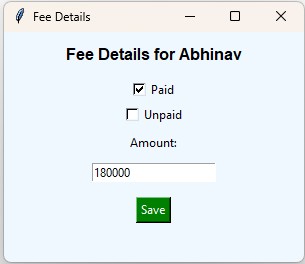
unpaid\_var.set(True) amount\_entry.insert(0, fee\_record.get("amount",""))

def save\_fee():

amount = amount\_entry.get().strip() if not amount or (not paid\_var.get() and not unpaid\_var.get()): messagebox.showerror("Error", "Please fill in all fields.")

return status = "Paid" if paid\_var.get() else "Unpaid" student["\_fee\_record"] = {"status": status, "amount": amount} messagebox.showinfo("Success", "Fee details saved successfully.") win.destroy()

tk.Button(win, text="Save", bg="green", fg="white", command=save\_fee).pack(pady=10)



### Faculty Salary Management

def manage\_faculty\_salaries(self):

win = tk.Toplevel(self.root) win.title("Manage Faculty Salaries") win.geometry("350x400") for faculty in faculty\_data:

name = faculty.get("Full Name") emp\_id = faculty.get("Employee ID") btn = tk.Button(win, text=f"{name} - {emp\_id}", width=40, command=lambda f=faculty: self.open\_salary\_details(f)) btn.pack(pady=3) def open\_salary\_details(self, faculty):

win = tk.Toplevel(self.root) win.title("Salary Details") win.geometry("300x230") win.configure(bg="aliceblue")

tk.Label(win, text=f"Salary Details for {faculty['Full Name']}", font=("Arial", 12, "bold"), bg="aliceblue").pack(pady=10) paid\_var = tk.BooleanVar() unpaid\_var = tk.BooleanVar() def on\_paid\_checked(): if paid\_var.get():

unpaid\_var.set(False) def on\_unpaid\_checked(): if unpaid\_var.get(): paid\_var.set(False)

tk.Checkbutton(win, text="Paid", variable=paid\_var, command=on\_paid\_checked, bg="aliceblue").pack()

tk.Checkbutton(win, text="Unpaid", variable=unpaid\_var, command=on\_unpaid\_checked, bg="aliceblue").pack() tk.Label(win, text="Amount:", bg="aliceblue").pack(pady=5) amount\_entry = tk.Entry(win) amount\_entry.pack(pady=5)

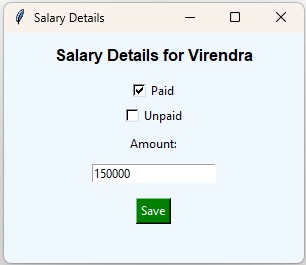
salary\_record = faculty.get("\_salary\_record", {}) if salary\_record: if salary\_record.get("status") == "Paid":

paid\_var.set(True) elif salary\_record.get("status") == "Unpaid":

unpaid\_var.set(True) amount\_entry.insert(0, salary\_record.get("amount","")) def save\_salary():

amount = amount\_entry.get().strip() if not amount or (not paid\_var.get() and not unpaid\_var.get()): messagebox.showerror("Error", "Please fill in all fields.")

return status = "Paid" if paid\_var.get() else "Unpaid" faculty["\_salary\_record"] = {"status": status, "amount": amount} messagebox.showinfo("Success", "Salary details saved successfully.") win.destroy() tk.Button(win, text="Save", bg="green", fg="white", command=save\_salary).pack(pady=10)



## Communication Portal Window Class (Main Interface)

class CommunicationPortalWindow:

def \_\_init\_\_(self, parent):

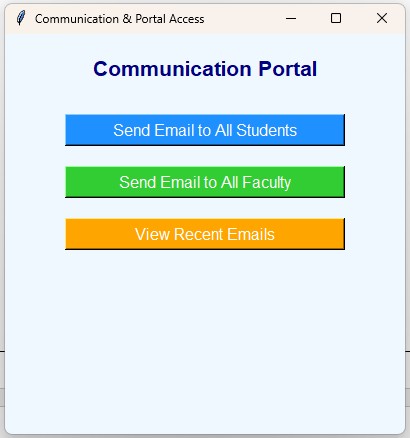
self.root = tk.Toplevel(parent) self.root.title("Communication & Portal Access") self.root.geometry("400x400") self.root.configure(bg="aliceblue")

tk.Label(self.root, text="Communication Portal", font=("Arial", 16, "bold"), bg="aliceblue", fg="navy").pack(pady=20)

tk.Button(self.root, text="Send Email to All Students", command=self.send\_email\_to\_students, bg="dodgerblue", fg="white", font=("Arial", 12), width=30).pack(pady=10)

tk.Button(self.root, text="Send Email to All Faculty", command=self.send\_email\_to\_faculty, bg="limegreen", fg="white", font=("Arial", 12), width=30).pack(pady=10)

tk.Button(self.root, text="View Recent Emails", command=self.view\_recent\_emails, bg="orange", fg="white", font=("Arial", 12), width=30).pack(pady=10)



### Email Composition & Sending

def send\_email\_to\_students(self):

self.open\_email\_window("students")

def send\_email\_to\_faculty(self):

self.open\_email\_window("faculty") def open\_email\_window(self, recipient\_type):

win = tk.Toplevel(self.root) win.title(f"Send Email to {recipient\_type.capitalize()}") win.geometry("450x350") win.configure(bg="aliceblue") tk.Label(win, text=f"Email By:", font=("Arial", 12), bg="aliceblue").pack(pady=5) email\_by\_entry = tk.Entry(win, font=("Arial", 12), width=40) email\_by\_entry.pack(pady=5)

tk.Label(win, text=f"Email to Branch (Leave blank for all {recipient\_type}):", font=("Arial", 12), bg="aliceblue").pack(pady=5) email\_to\_branch\_entry = tk.Entry(win, font=("Arial", 12), width=40) email\_to\_branch\_entry.pack(pady=5) tk.Label(win, text="Write Email:", font=("Arial", 12), bg="aliceblue").pack(pady=5) email\_content\_entry = tk.Text(win, font=("Arial", 12), height=8, width=40) email\_content\_entry.pack(pady=5) message\_label = tk.Label(win, text="", font=("Arial", 11, "bold"), fg="red", bg="aliceblue") message\_label.pack(pady=3)

def send\_email():

email\_by = email\_by\_entry.get().strip() branch = email\_to\_branch\_entry.get().strip() email\_content = email\_content\_entry.get("1.0", tk.END).strip() if not email\_by or not email\_content:

message\_label.config(text="Warning: 'Email By' and 'Write Email' fields are mandatory!")

return

if recipient\_type == "students":

recipients = self.filter\_students\_by\_branch(branch) recipient\_str = f"All Students" if not branch else f"Students in branch '{branch}'"

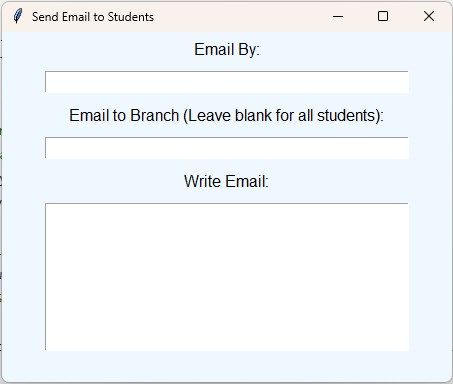
else: recipients = self.filter\_faculty\_by\_branch(branch) recipient\_str = f"All Faculty" if not branch else f"Faculty in branch '{branch}'" if not recipients:

message\_label.config(text=f"Warning: No {recipient\_type} found for the specified branch.")

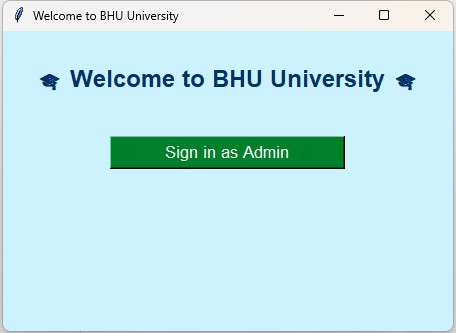
return

recent\_emails.append({ "by": email\_by,

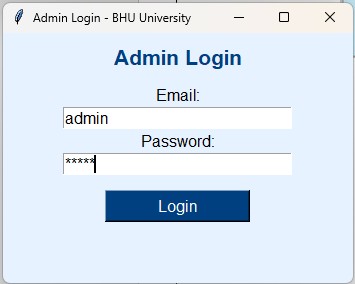
"to": recipient\_str, "content": email\_content}) message\_label.config(text="Email sent successfully!", fg="green") email\_by\_entry.delete(0, tk.END) email\_to\_branch\_entry.delete(0, tk.END) email\_content\_entry.delete("1.0", tk.END) tk.Button(win, text="Send", bg="green", fg="white", font=("Arial", 12), command=send\_email).pack(pady=10)



# OUTPUT

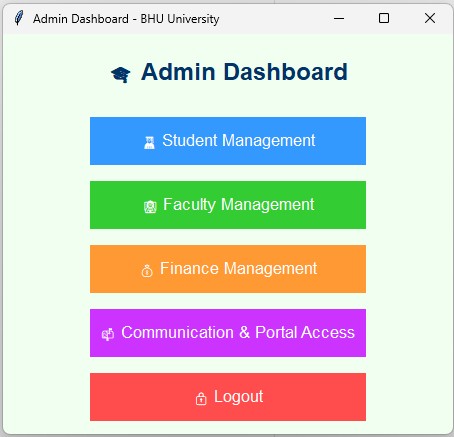


Now we click on sign in as Admin

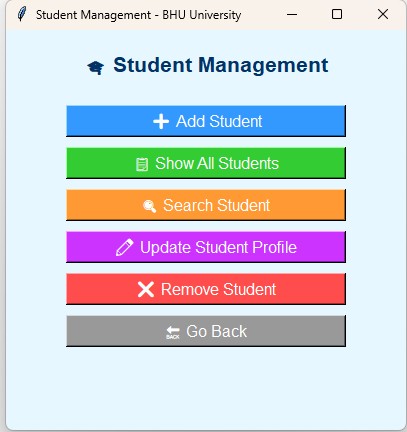


Login with username-admin & password-12345

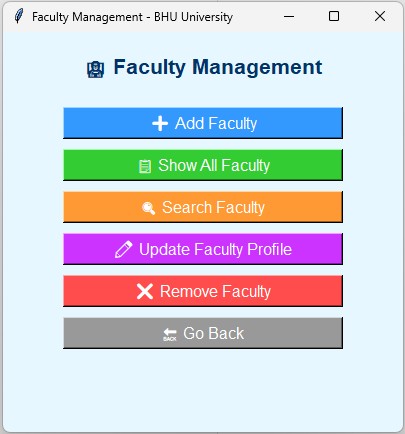
Now you move to Admin Dashboard



Student management system



Faculty management system



Finance management

