Based on this sheet I want to develop a VS Code Project for Computer Faculties Salary Budget Demand on monthly basis in python and html and css that I can deploy on streamlit. Also based on Joining Date of CF in ICT his or her salary is incremented at some rate on basic salary. Help me develop this project that helps the ICT Coordinators to prepare budget demand for school cfs school's bank account.

That’s a fantastic initiative, Jagdev—this kind of tool could really streamline budget planning for ICT coordinators and make salary forecasting more transparent and scalable. Let’s break this down into a modular, maintainable Streamlit project that aligns with your pedagogical and technical style.

**🧠 Project Overview: “CF Salary Budget Demand Generator”**

**🎯 Goals**

* Generate monthly salary budgets for Computer Faculties (CFs) school-wise.
* Apply salary increments based on Date of Joining (DoJ).
* Display totals per school and allow export/download.
* Empower ICT Coordinators with a clean, interactive dashboard.

**🗂️ Project Structure**

plaintext

cf\_salary\_budget/

├── app.py

├── data/

│ └── cf\_salary\_details.xlsx

├── modules/

│ ├── salary\_calculator.py

│ ├── data\_loader.py

│ └── utils.py

├── assets/

│ ├── styles.css

│ └── logo.png

├── templates/

│ └── index.html

└── README.md

**🧮 Salary Calculation Logic**

**🔁 Increment Rule (example logic)**

* **Base increment rate**: 3% annually on Basic Salary.
* **Effective increment** = Years of Service × Increment Rate × Basic Salary
* **Years of Service** = Current Date - Date of Joining

python

def calculate\_incremented\_salary(join\_date, basic\_salary, rate=0.03):

years = (pd.Timestamp.today() - pd.to\_datetime(join\_date)).days // 365

increment = basic\_salary \* rate \* years

return basic\_salary + increment

**🧑‍💻 app.py (Streamlit Frontend)**

python

import streamlit as st

from modules.data\_loader import load\_cf\_data

from modules.salary\_calculator import compute\_school\_budgets

st.set\_page\_config(page\_title="CF Salary Budget Demand", layout="wide")

st.title("📊 Computer Faculty Salary Budget Demand Generator")

df = load\_cf\_data("data/cf\_salary\_details.xlsx")

budget\_df = compute\_school\_budgets(df)

st.dataframe(budget\_df)

if st.button("Download Budget Summary"):

st.download\_button("Download CSV", budget\_df.to\_csv(index=False), "cf\_budget\_summary.csv")

**🧩 salary\_calculator.py**

python

def compute\_school\_budgets(df):

df["Incremented Basic"] = df.apply(

lambda row: calculate\_incremented\_salary(row["Date of Joining in the ICT"], row["Basic Salary"]), axis=1

)

df["Total Salary"] = df["Incremented Basic"] + df[["DA @ 181 %", "HRA @ 10%", "RA @ 6%", "CCA",

"Border Allowance", "Handicap Allowance",

"Medical Allowance", "Mobile Allowance"]].sum(axis=1)

school\_budgets = df.groupby(["Name of School", "Bank Account No, (State Bank Of India only)"])["Total Salary"].sum().reset\_index()

school\_budgets.rename(columns={"Total Salary": "Monthly Budget Demand"}, inplace=True)

return school\_budgets

**🎨 Styling (styles.css)**

css

body {

font-family: 'Segoe UI', sans-serif;

background-color: #f9f9f9;

}

h1 {

color: #2c3e50;

}

**🌐 Deployment Tips**

* Use streamlit run app.py locally.
* For public access, deploy via Streamlit Community Cloud.
* Add .streamlit/config.toml for custom theming if needed.

**🔮 Next Steps**

Would you like help scaffolding the actual Python modules or setting up the Streamlit deployment? We can also add features like:

* PDF export per school
* Role-based access (ICT Coordinator vs Admin)
* Historical budget tracking
* GitHub Actions for auto-deployment

Let’s build this into something that’s not just functional—but empowering.