VIT®

BHOPAL

CSE PROJECT

VITYARTHI

Menstrual Cycle Prediction Tool

NAME- NAVDHA JAIN

DATE- 25-11-25

REGISTRATION NUMBER - 25BAI11528

INTRODUCTION

Many individuals experience difficulty in accurately predicting their menstrual cycles. This unpredictability often causes inconvenience in planning daily activities, travel, and managing healthcare needs. Most available tools are either too complex, inaccessible, or unreliable for everyday use.

This project aims to address this gap by providing a simple, reliable, and accessible system that uses users' historical menstrual cycle data to estimate the likely start date of their next period. By analyzing previous period start dates, the system calculates the average length between cycles and then forecasts the upcoming cycle date.

Such a tool can empower users with better planning capabilities, raise awareness about their menstrual health, and reduce the uncertainty around cycle timing. This not only helps individuals in their daily life management but can also encourage greater attentiveness to reproductive health

PROBLEM STATEMENT

Many individuals face challenges in accurately predicting their menstrual cycles, which can disrupt planning for daily activities, travel, and healthcare. Most existing tools are either too complex or generalized, lacking personalization and ease of use. As a result, many people remain uncertain about when to expect their next period, causing stress and inconvenience.

This project aims to develop a simple, user-friendly system that leverages personal menstrual cycle history to provide reliable predictions of the next period start date. By analyzing past cycle data and calculating an average cycle length, the tool helps users anticipate their menstrual cycle timing more effectively.

With better predictions, users can plan their lives with greater confidence, manage menstrual health proactively, and reduce anxiety related to cycle unpredictability. This straightforward data-driven approach empowers users to monitor their reproductive health conveniently and encourages greater awareness of cycle patterns and irregularities.

OBJECTIVES

- To develop a tool that accurately tracks menstrual cycle start and end dates, enabling users to maintain a personal cycle history.
- To calculate the length of each menstrual cycle by analyzing the start dates, allowing for an understanding of individual cycle variability.
- To provide a data-driven and personalized prediction of the most likely start date for the next menstrual period based on recorded cycle lengths.
- To support users in their personal planning, helping them manage activities, appointments, and healthcare needs more effectively.
- To raise awareness and promote better menstrual health by enabling users to observe patterns and irregularities in their cycles over time.
- To create a simple and user-friendly interface that can be easily used by individuals, including those with limited technical experience.

FUNCTIONAL REQUIREMENTS

- Allow users to input previous menstrual cycle start and end dates accurately and easily.
- Store cycle data efficiently for retrieval and historical analysis.
- Calculate cycle lengths by measuring the interval between the start dates of consecutive cycles.
- Compute the average cycle length based on recorded historical data to reflect individual cycle patterns.
- Predict the next period start date using the calculated average cycle length, providing an estimated date customized to the user.
- Display a clear, user-friendly overview of the predicted next period and the history of previous cycles.
- Handle scenarios with limited or irregular cycle data gracefully, providing meaningful feedback or prompts to the user.
- Ensure data privacy and security if integrated with persistent storage or external systems.
- Provide console-based interaction with simple input and output handling to accommodate various user skill levels.
- Allow easy modification or expansion of the application to add future features such as symptom tracking or cycle irregularity alerts.

NON FUNCTIONAL REQUIREMENTS

- ***Accuracy***: The system's predictions must closely reflect the user's actual cycle history, reducing discrepancies over time with consistent and correct data input.
- ***Usability***: The user interface should be intuitive, simple, and accessible to users of all technical skill levels. Clear instructions, minimal input requirements, and straightforward navigation are essential.
- ***Reliability***: The system should handle cases with incomplete, irregular, or missing data gracefully, providing useful feedback or estimations without crashing or producing erroneous results.
- ***Privacy & Data Security***: All user data must be stored securely, respecting privacy standards. When implemented on a larger scale, encryption, secure storage, and anonymization mechanisms should be employed to protect sensitive health data.
- ***Performance***: The system should respond quickly to user inputs and predictions, with minimal delays, even with large datasets or multiple users (if expanded to multi-user support).
- ***Maintainability***: Codebase should be modular, well-documented, and easy to update or extend, facilitating future feature additions or improvements.
- ***Compatibility***: The application should work across different environments (e.g., Windows, macOS, Linux) and be adaptable for mobile or web interfaces if scaled later.
- ***Scalability***: As user adoption grows, the architecture should support increased data storage, user management, and feature

integration without significant performance drops.

SYSTEM ARCHITECTURE

Overview

The menstrual cycle prediction tool features a simple and modular architecture, designed to collect user data, perform cycle calculations, and predict the next period.

Components

- User Interface (UI):
  A command-line interface for users to input cycle start/end dates and view predictions. This can be upgraded to graphical or web-based interfaces later.
- Data Storage:
  Temporarily stores cycle data during runtime in data structures, with future options to connect to databases for persistence.
- Calculation Engine:
  Computes cycle lengths from start date differences and calculates the average cycle length, handling irregular or incomplete data smoothly.
- Prediction Module:
  Uses the average cycle length to estimate the next period start date through simple date arithmetic.
- Output Module:
  Displays predictions and cycle history clearly to users, supporting possible future visualizations like charts.

Data Flow

User Input → Data Storage → Calculation → Prediction → User Output

Extensibility

Modular design supports future upgrades including database integration, web/mobile UIs, and advanced prediction algorithms.

DESIGN DIAGRAMS

Use Case Diagram

- Illustrates interactions between the user and the system.
- Main use cases:
  - Input cycle data (start/end dates)
  - View calculated cycle lengths
  - Predict next period start date
  - View prediction and historical overview

Class Diagram

- Key classes:
  - CycleData: stores start and end dates
  - CycleCalculator: calculates cycle lengths and averages
  - PeriodPredictor: uses average length to predict next period date
  - UserInterface: manages input and output interactions

Sequence Diagram

- Sequence shows the flow:
  - User inputs cycle data → System stores data
  - Calculation engine processes data → Average cycle length calculated
  - Prediction module estimates next period start → Result displayed to user

Flowchart / Workflow Diagram

- Steps:
  1. Input cycle data
  2. Validate data
  3. Calculate cycle lengths
  4. Compute average length
  5. Predict next period
  6. Display output

DESIGN DECISIONS & RATIONALE

Simplicity and Accessibility

- The project uses a command-line interface to keep the tool simple and accessible to users with basic technical skills.
- Avoids complex UI components to focus on core functionality.

Data-Driven Approach

- Uses historical cycle start dates as the primary input, enabling personalized and data-driven predictions.
- Average cycle length is chosen as a straightforward and effective method for predicting the next period, reflecting common menstrual cycle patterns.

Modular Design

- Separates concerns into modules: data storage, calculation engine, prediction, and user interface.
- Facilitates maintenance and future enhancements, such as adding databases or enhanced prediction algorithms.

Handle Data Variability

- Designed to work with real-world cycle data that may be irregular or incomplete.
- Implements error handling and prompts to ensure robustness and user guidance.

Privacy and Security Considerations

- The current implementation uses in-memory data storage for simplicity.
- Emphasizes need for secure storage and privacy protections if expanded to persistent or multi-user systems.

Future-Proofing

- Architecture and implementation allow integration of machine learning models for improved accuracy in future versions.
- Easily extensible to mobile or web applications for broader accessibility.

IMPLEMENTATION DETAILS

Programming Language & Environment

- Implemented in Python, leveraging built-in libraries like datetime for date calculations.
- Uses simple data structures (lists, dictionaries) to hold cycle data in memory for easy manipulation and calculation.

Core Algorithm

- Calculates the difference in days between consecutive cycle start dates to determine cycle lengths.
- Computes the average of these cycle lengths to estimate the length of a typical menstrual cycle for the user.
- Predicts the next period start date by adding the average cycle length to the last recorded start date.

Input & Output

- Users input previous cycle start and end dates directly in the code (cycle_data list).
- Outputs include recorded cycle lengths and the predicted next start date printed to the console.

Data Handling

- Designed to handle incomplete or irregular data by calculating averages only when sufficient data is available.
- Future scope to include persistent storage (database) or user input interfaces for broader usability.

Extensibility & Improvements

- Modular code structure enables easy extension to include more sophisticated prediction models.
- Machine learning algorithms can be integrated later to improve prediction accuracy.

- Interface upgrades from CLI to graphical or web-based frontends are possible for enhanced user experience.

SCREENSHOTS / RESULTS

```python
from datetime import date, timedelta

# In a real application, this data would be stored in a database
cycle_data = [
    {"start": date(2025, 8, 1), "end": date(2025, 8, 5)},
    {"start": date(2025, 8, 29), "end": date(2025, 9, 3)},
    {"start": date(2025, 9, 27), "end": date(2025, 10, 1)},
    {"start": date(2025, 10, 25), "end": date(2025, 10, 29)},
]

def calculate_cycle_lengths(data):
    """Calculates the duration of each tracked cycle."""
    lengths = []
    for i in range(len(data) - 1):
        # Days from the start of the current cycle to the start of the next
        length = (data[i+1]["start"] - data[i]["start"]).days
        lengths.append(length)
    return lengths

def predict_next_period(data):
    """Predicts the start date of the next period based on average cycle length."""
    if len(data) < 2:
        return "Need at least two cycles to predict."
    lengths = calculate_cycle_lengths(data)
    average_cycle_length = sum(lengths) / len(lengths)
    last_period_start = data[-1]["start"]
    # Add the average length to the last start date to predict the next start
    next_start_date = last_period_start + timedelta(days=round(average_cycle_length))
    return next_start_date

# --- Usage ---
cycle_lengths = calculate_cycle_lengths(cycle_data)
next_period_prediction = predict_next_period(cycle_data)

print(f"Recorded cycle lengths: {cycle_lengths} days")
print(f"Next period predicted start date: {next_period_prediction}")
```

```
Recorded cycle lengths: [28, 29, 28] days
Next period predicted start date: 2025-11-22
```

TESTING APPROACH

Unit Testing

- Test individual functions that calculate cycle lengths and averages.
- Verify date arithmetic logic for accuracy in days calculations.
- Validate prediction logic by comparing expected vs. actual output for given cycle data.

Integration Testing

- Test data flow between input, storage, calculation, and prediction modules.
- Ensure output consistency when combining modules.

Boundary Testing

- Test edge cases such as cycles with minimum data points (e.g., 2 cycles).
- Handle irregular cycles and missing data entries gracefully.

Manual Testing

- Use sample and real user cycle data for functional verification.
- Confirm correct display of cycle lengths and predicted next period date.

Performance Testing

- Check performance with increasing amount of cycle data.
- Ensure prompt prediction generation without delays.

Usability Testing

- Confirm the interface is intuitive and inputs are correctly received.
- Evaluate error messages and feedback for users with incomplete data.

Future Automated Testing

- Recommend integration with automated testing frameworks for regression testing.
- Plan to include tests for enhanced UI or machine learning models as features expand.

CHALLENGES FACED

- **Irregular Cycle Data**:
  Variability and irregularities in menstrual cycles make prediction less straightforward. Handling inconsistent cycle lengths is complex.
- **Incomplete or Missing Data**:
  Users may forget to record cycle dates or input incomplete information, leading to gaps that affect prediction accuracy.
- **Accuracy of Predictions:**
  Simple averaging methods may not capture nuances in individual cycles. Advanced modeling is needed to improve reliability.
- **User Adherence:**
  Maintaining consistent and accurate tracking over time is challenging but crucial for meaningful predictions.
- **Data Privacy:**
  Protecting sensitive health data while providing useful insights requires careful design and secure data handling.
- **Limited Interface:**
  A command-line interface, while simple, may reduce usability for non-technical users compared to graphical or mobile apps.
- **Scalability for Future Enhancements:**
  Designing for easy integration of machine learning or multi-user support poses architectural challenges.

LEARNINGS & KEY TAKEAWAYS

- **Data Quality is Crucial:** Accurate and consistent cycle data input is essential for reliable predictions.
- **Simplicity Improves Usability:** Keeping the interface simple ensures that users of varying technical skills can benefit.
- **Personalization Matters:** Tailoring predictions based on individual cycle history enhances usefulness and accuracy.
- **Handling Irregularities:** Effective prediction systems must accommodate irregular and incomplete data gracefully.
- **Modular Design Enables Flexibility:** Separating functions into modules allows for easier maintenance and future expansions.
- **Scope for Advanced Techniques:** While averages work well initially, integrating machine learning could improve prediction over time.
- **Privacy is Paramount:** Even simple tools must consider user data protection and privacy from the outset.
- **User Engagement is Key:** Encouraging consistent user input increases data reliability and prediction quality.

FUTURE ENHANCEMENTS

- **Graphical User Interface (GUI):**
  Develop a user-friendly GUI or mobile app for easier data input and visualization.
- **Persistent Storage:**
  Integrate databases to store cycle data securely for long-term tracking and analysis.
- **Advanced Prediction Models:**
  Incorporate machine learning algorithms to improve prediction accuracy, especially for irregular cycles.
- **Symptom and Mood Tracking:**
  Enable users to log symptoms, moods, and other health data for comprehensive menstrual health insights.
- **Personalized Alerts & Notifications:**
  Add reminders for upcoming periods, fertile windows, and ovulation days.
- **Multi-user Support:**
  Allow multiple user profiles with secure data segregation.
- **Data Export & Reporting:**
  Provide options to export cycle data and generate periodic health reports.
- **Privacy Enhancements:**
  Implement encryption and privacy controls to protect sensitive user data.
- **Integration with Wearable Devices:**
  Sync with devices like smartwatches for automated data collection.

REFERENCES

This section includes citations for all key resources utilized during the development of the menstrual cycle prediction tool. Common reference categories include:

- **Libraries:** Official documentation for Python libraries such as datetime, and any other modules used for date calculations and data handling.
- **Tutorials and Guides**: Authoritative tutorials from resources like Real Python and official Python documentation that provided step-by-step guidance for implementing cycle tracking and prediction logic.
- **Sample Datasets:** Any synthetic or publicly available menstrual cycle data used to test and validate prediction algorithms.
- **API Documentation:** Documentation for any external tools or frameworks referenced, including Python standard library references for data manipulation and input/output operations.
- **Research Papers or Articles:** Key academic sources and articles on menstrual cycle prediction and modeling, such as:
  - Predictive Modeling of Menstrual Cycle Length (arXiv, 2023)
  - The forecasting of menstruation based on a state-space model (Statistics in Medicine, 2017)
  - A predictive model for next menstrual cycle start date (JAMIA, 2021)
  - Menstrual Cycles Prediction using Artificial Neural Network (IEEE, 2023)
  - Real-life insights on menstrual cycles and ovulation using big data (Human Reproduction Open, 2020)
- **Development Tools:** Documentation for tools used in coding, testing, and deployment (e.g., Python interpreter, terminal/CLI, code editors).