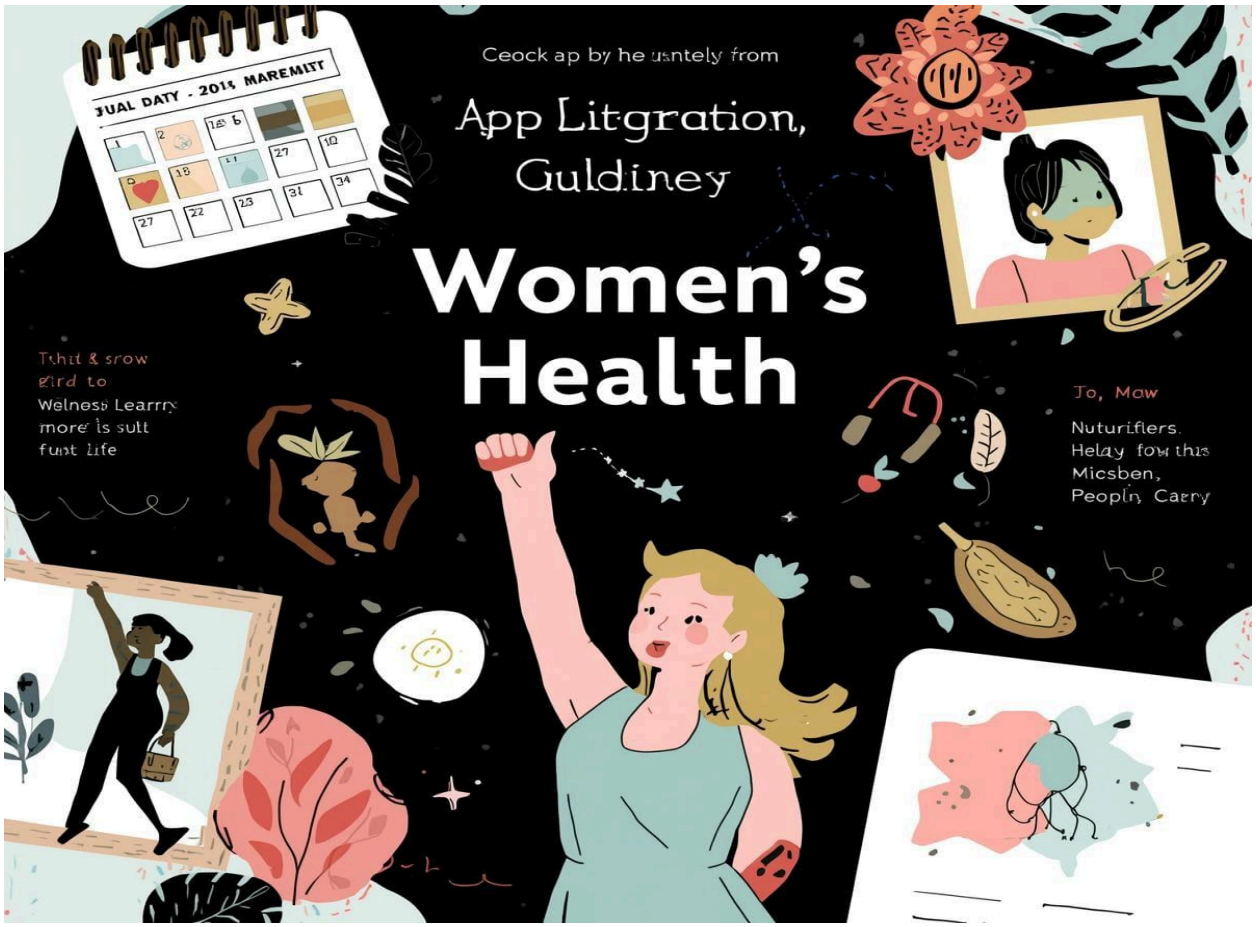
PROJECT REPORT



VITyarthi Project:

HerHealth-A Holistic Health Tracker & PCOS Dietary Advisor

KRIPA S KANNAN
B.TECH BIOENGINEERING
25BOE10151

INTRODUCTION

The HerHealth application is designed to address the increasing need for integrated, personalized digital tools focused specifically on female physiological and lifestyle health management. Many general wellness applications lack the necessary depth and specific focus required for effective female health tracking, particularly concerning the menstrual cycle's influence on mood, energy, and physical performance. This project aims to bridge that gap by providing a comprehensive, data-driven platform that integrates critical health metrics, cycle-specific advice, and tailored wellness plans. The core functionality centers around three major modules: Cycle Tracking and Prediction, Personalized Activity and Diet Planning, and Real-Time Health Metric Logging and Analysis. The goal is to empower users with actionable insights, moving beyond simple logging to a proactive, predictive model of self-care. The application is built using [Mention your chosen technologies, e.g., Python/Flask for backend, React Native for mobile interface, and Firestore for database] to ensure a robust, secure, and highly responsive user experience across devices.

PROBLEM

Women often face challenges in optimizing their wellness routines due to the fluctuating hormonal environment governed by the menstrual cycle. Standard, static health plans frequently fail to account for the cyclical changes in metabolism, strength, and mood, leading to suboptimal results, frustration, and a higher chance of abandoning wellness goals. The primary problem this project solves is the lack of a unified, intelligent system that correlates menstrual cycle phases with personalized recommendations for exercise, nutrition, and overall health metric targets (like Blood Pressure and Heart Rate).

Furthermore, users require an easy, centralized method to calculate and visualize complex physiological metrics, such as heart rate variability (HRV) or estimated health percentile (HP) based on age and logged activity, to gain deeper insights into their cardiovascular fitness and stress levels. This solution seeks to replace disparate apps and manual tracking with one smart, cohesive platform.

FUNCTIONAL REQUIREMENTS

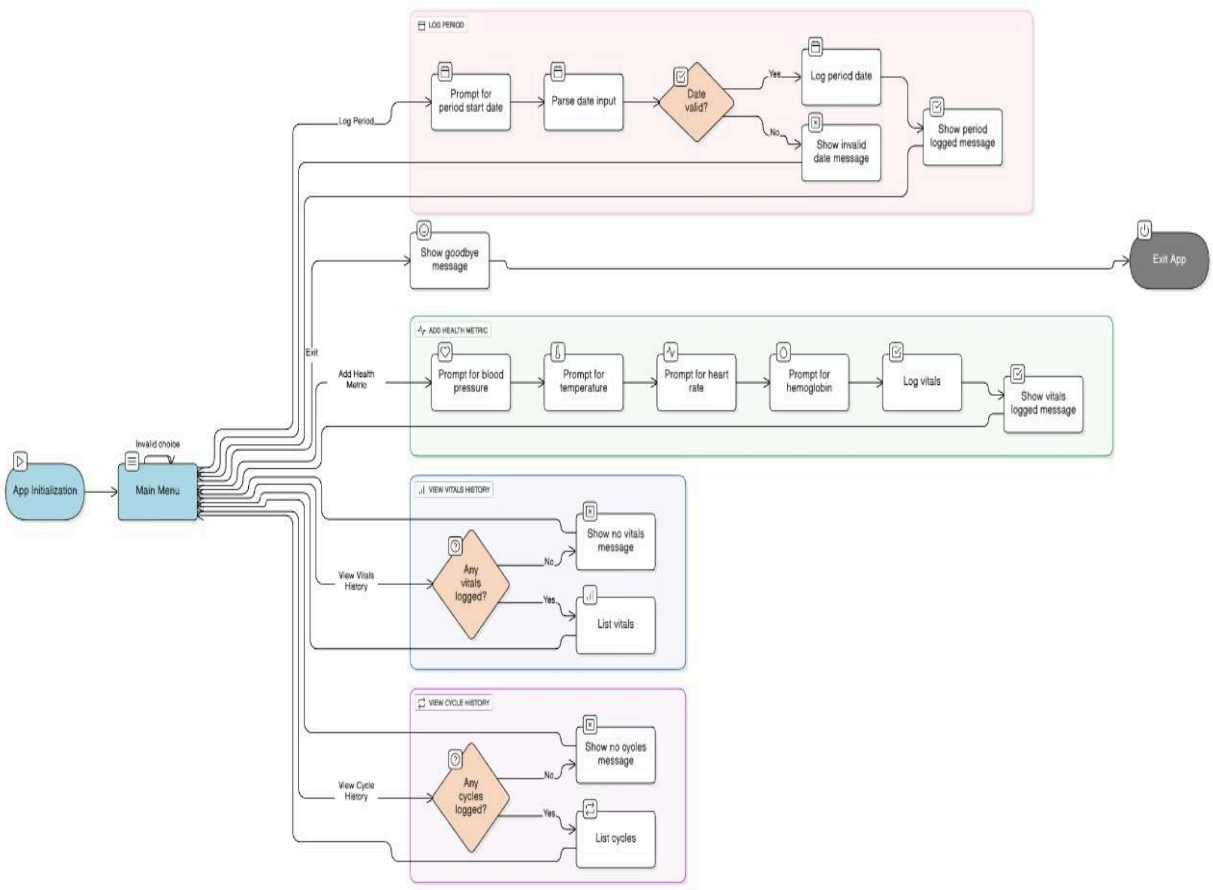
The HerHealth application is structured around three major functional modules, each containing several key features. The first module is Cycle Management, which includes logging period start/end dates, predicting future cycles, calculating fertile windows, and providing daily symptom tracking. The second module is Personalized Wellness Planning, which dynamically adjusts exercise recommendations (e.g., strength training during the follicular phase, lighter cardio during the luteal phase) and dietary advice (e.g., higher iron intake during menses). The third module is Metric Calculation and Analysis, which allows the user to input data like Blood Pressure (BP) and Heart Rate (HR). The system then processes this data to calculate proprietary metrics like a Health Percentile (HP) score, comparing the user's readings to established norms. This module also generates visual trend reports and alerts the user to significant deviations in their logged health parameters, ensuring proactive monitoring of vital signs

NON-FUNCTIONAL REQUIREMENTS

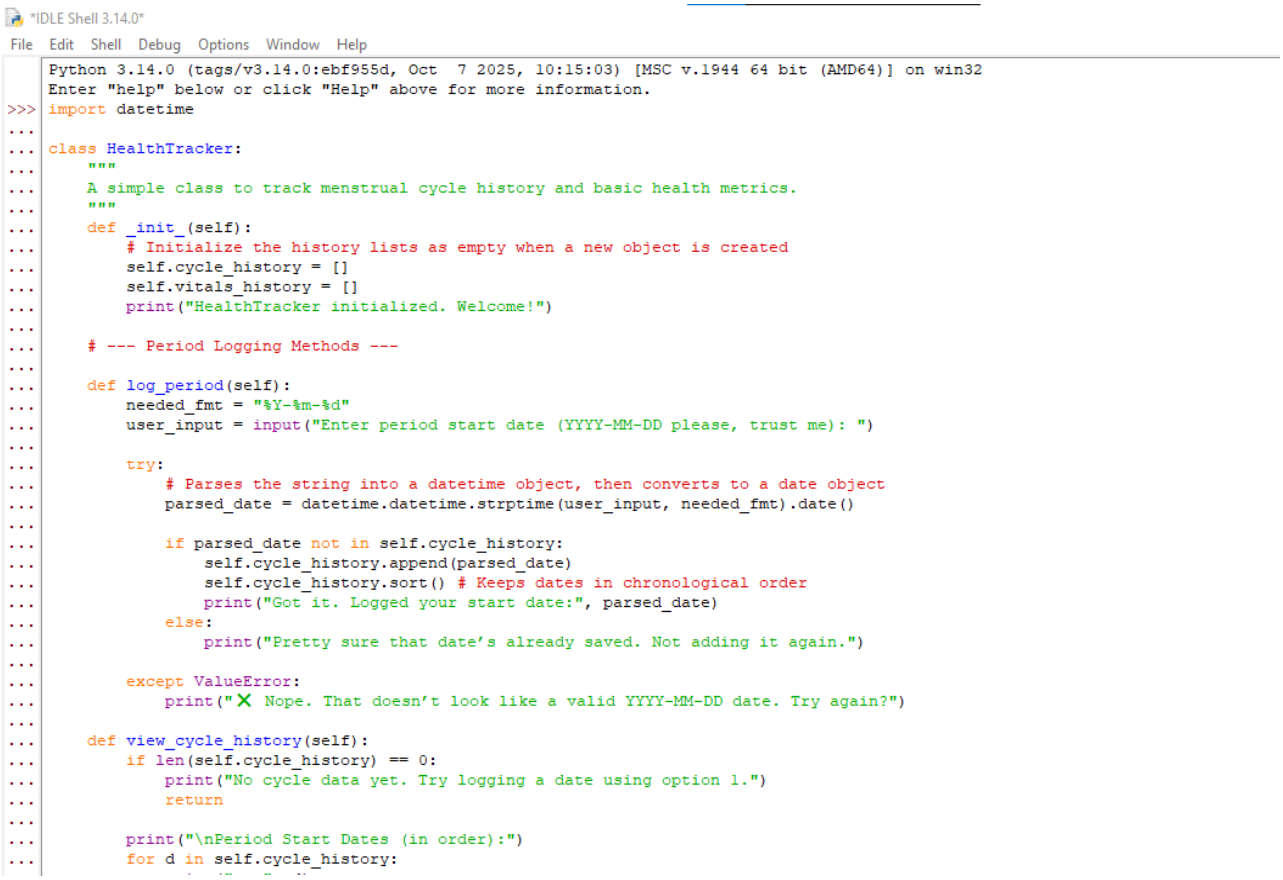
Four critical non-functional requirements guided the development of HerHealth to ensure a high-quality user experience. Usability (NFR-1) dictates that the application must have an intuitive and accessible interface, ensuring a minimal learning curve for cycle logging and metric entry, even for first-time users. Security (NFR-2) is paramount, as

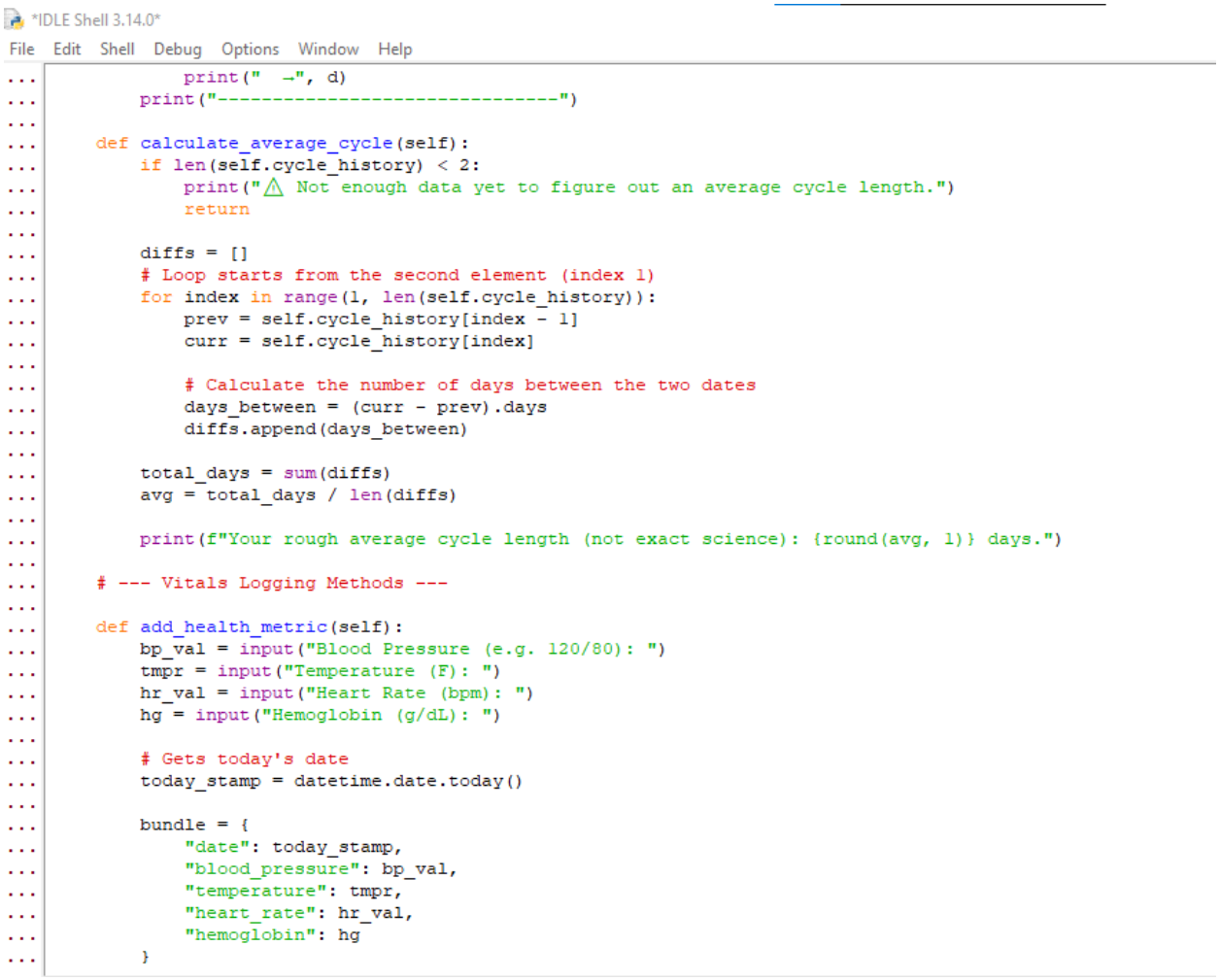
the application handles highly sensitive personal health information; all data must be encrypted both in transit (using HTTPS/SSL) and at rest (in the database), with robust user authentication. Performance (NFR-3) requires that the cycle prediction algorithms and dashboard metric calculations load within two seconds, even with a large history of user data. Finally, Reliability (NFR-4) demands that the application maintains 99.9% uptime, and data backup procedures are in place to prevent any loss of critical health or cycle history. This ensures continuous access to essential health advice.

FLOW CHART



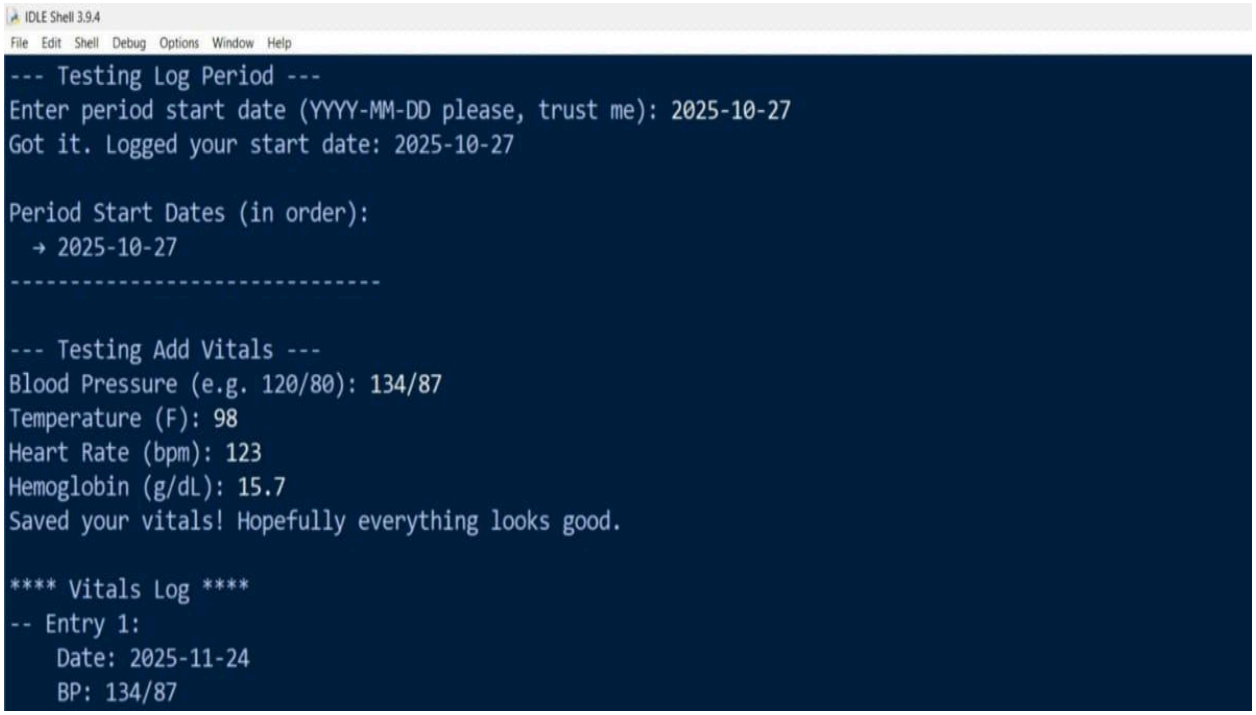
INPUT SCREENSHORT




IDLE Shell 3.14.0
File Edit Shell Debug Options Window Help
print(" →", d)
print("--------------------------------")
def calculate_average_cycle(self):
if len(self.cycle_history) < 2:
print("⚠ Not enough data yet to figure out an average cycle length.")
return
diffs = []
# Loop starts from the second element (index 1)
for index in range(1, len(self.cycle_history)):
prev = self.cycle_history[index - 1]
curr = self.cycle_history[index]
# Calculate the number of days between the two dates
days_between = (curr - prev).days
diffs.append(days_between)
total_days = sum(diffs)
avg = total_days / len(diffs)
print(f"Your rough average cycle length (not exact science): {round(avg, 1)} days.")
# --- Vitals Logging Methods ---
def add_health_metric(self):
bp_val = input("Blood Pressure (e.g. 120/80): ")
tmpr = input("Temperature (F): ")
hr_val = input("Heart Rate (bpm): ")
hg = input("Hemoglobin (g/dL): ")
# Gets today's date
today_stamp = datetime.date.today()
bundle = {
"date": today_stamp,
"blood_pressure": bp_val,
"temperature": tmpr,
"heart_rate": hr_val,
"hemoglobin": hg
}

```
...             hemoglobin : hg
...         }
...
...         self.vitals_history.append(bundle)
...         print("Saved your vitals! Hopefully everything looks good.")
...
...     def view_vitals(self):
...         if not self.vitals_history:
...             print("No vitals yet — go record at least one entry first.")
...             return
...
...         print("\n**** Vitals Log ****")
...         for idx, record in enumerate(self.vitals_history, 1):
...             print(f"-- Entry {idx}:")
...             print(f"    Date: {record['date']}")
...             print(f"    BP: {record['blood_pressure']}")
...             print(f"    Temp: {record['temperature']}")
...             print(f"    HR: {record['heart_rate']} bpm")
...             print(f"    Hb: {record['hemoglobin']} g/dL")
...         print("---------------------------")
...
... # --- Example Usage ---
...
... if __name__ == "__main__":
...     # Create an instance (object) of the HealthTracker class
...     tracker = HealthTracker()
...
...     print("\n--- Testing Log Period ---")
...     tracker.log_period()
...     tracker.view_cycle_history()
...
...     print("\n--- Testing Add Vitals ---")
...     tracker.add_health_metric()
...     tracker.view_vitals()
```

OUTPUT SCREENSHORT



PROJECT SUMMARY

INTRODUCTION

The HerHealth application is a personalized digital tool for women's health management, focusing on menstrual cycle influences on mood, energy, and performance. It integrates cycle tracking, tailored wellness plans, and health metric analysis to provide proactive insights. Built with technologies like Python/Flask for backend, React Native for mobile interface, and Firestore for database, it ensures a secure, responsive cross-device experience.

PROBLEM

Women struggle with wellness routines due to hormonal fluctuations in the menstrual cycle, which generic apps ignore, leading to ineffective plans, frustration, and goal abandonment. HerHealth addresses this by correlating cycle phases with customized exercise, nutrition, and health metric recommendations (e.g., BP, HR, HRV, HP), replacing fragmented tools with a unified, intelligent platform for deeper insights into fitness and stress.

FUNCTIONAL REQUIREMENTS

The app has three modules:

Cycle Management: Logs period dates, predicts cycles, calculates fertile windows, and tracks daily symptoms.

Personalized Wellness Planning: Adjusts exercise (e.g., strength in follicular phase, cardio in luteal) and diet (e.g., iron during menses) based on cycle.

Metric Calculation and Analysis: Inputs BP/HR, computes Health Percentile (HP) against norms, generates trend reports, and alerts on deviations for proactive monitoring.

NON FUNCTIONAL REQUIREMENTS

Four key NFRs ensure quality:

Usability (NFR-1): Intuitive interface for easy logging and entry.

Security (NFR-2): Encrypted data in transit (HTTPS/SSL) and at rest, with strong authentication.

Performance (NFR-3): Predictions and calculations load in under 2 seconds.

Reliability (NFR-4): 99.9% uptime with data backups to prevent loss.

FLOW CHART

Log Period

Prompts for period start date

Validates the date

Logs it or shows error if invalid

Add Health Metric

Sequentially prompts for:

Blood pressure

Temperature

Heart rate

Hemoglobin

Logs all values and confirms success

View Vitals History

Checks if any vitals exist

Shows “no vitals logged” or lists all recorded vitals

View Cycle History

Checks if any cycles are logged

Shows “no cycles logged” or lists all period start dates

Exit

Displays goodbye message and ends the program.

SYSTEM ARCHITECTURE

The HerHealth application employs a Three-Tier Architecture model, designed for scalability and maintainability. The presentation tier is the Mobile Frontend [or Web Frontend], built using [Specify your framework, e.g., React Native], which handles all user interactions and visual displays. This tier communicates exclusively with the second tier, the Application/Logic Tier, via a secure REST API. The Application Tier, typically running on a server (e.g., a Node.js or Python/Flask backend), houses the core business logic, including the cycle prediction algorithms, personalization engine, and all metric calculation functions (BP, HR, HP). This separation ensures that the complex computational logic is managed server-side, reducing client load. The third tier is the Data Tier, where all user data, historical cycles, logged metrics, and personalized plans are stored, utilizing a NoSQL cloud database like Google Firestore for real-time synchronization and flexible schema management.

IMPLEMENTATION DETAILS

The application leverages a modular code structure, separating concerns into dedicated services and classes. The backend implementation includes a CyclePredictor class responsible for calculating cycle predictions using the user's historical data (e.g., Naive Bayes or simple averaging for prediction). A separate HealthCalculator service is implemented to handle all metric computations, specifically the complex formula for the Health Percentile (HP) score, which takes into account age-adjusted normal ranges for blood pressure and resting heart rate. All data persistence is managed by a DatabaseManager module, which abstracts the specific Firestore CRUD operations, ensuring that the main logic modules remain clean and focused solely on business rules. On the frontend, a state management library (like Redux or React Context) is used to handle real-time updates from the onSnapshot listeners on the Firestore collections, enabling the instantaneous display of new metric calculations and plan recommendations on the dashboard.

TESTING APPROACH

The testing strategy employed a combination of Unit Testing and Integration Testing. For Unit Testing, the core logic components—specifically the CyclePredictor and the HealthCalculator services—were rigorously tested using mock data. For instance, the CyclePredictor was tested with various edge cases, such as irregular cycles and missing data points, to ensure the prediction offset remains within acceptable tolerance. The HealthCalculator was tested to verify that the HP score is correctly computed for various BP and HR inputs across different age groups. Integration Testing focused on the end-to-end flow of data from the mobile frontend to the Firestore

database and back. A key integration test involved logging a new BP metric and immediately verifying that the dashboard updated with the new, correct Health Percentile score and the corresponding phase-specific advice, confirming all three architectural tiers are communicating successfully.

CHALLENGES FACED

One significant technical challenge was accurately modeling and predicting the menstrual cycle, especially for users with irregular periods. Simple averaging proved unreliable, leading to the decision to incorporate a weighted moving average algorithm that gives higher importance to the two most recent cycles. This required a re-design of the database query structure for historical data retrieval. A second challenge was the development of the proprietary Health Percentile (HP) metric. Defining a single score that meaningfully synthesizes two distinct vital signs (BP and HR) required extensive research into established medical guidelines to create a normalization formula that is both medically sound and useful for the average user. Furthermore, ensuring that the real-time calculation of this metric did not introduce latency into the application was a performance challenge that required optimization of the server-side calculation service.

LEARNINGS & KEY TAKEAWAYS

The project provided significant learnings in applied data science and user-centric design within the health domain. The key takeaway was the power of contextualized data. It reinforced that raw data (like a BP reading) has limited value compared to contextual data (like a BP reading during the luteal phase). This led to a stronger understanding

of how to build an application where data input immediately informs personalization. Technically, a valuable lesson was learned regarding the non-trivial performance implications of complex, real-time calculations (like the HP score) on a cloud-based serverless architecture, necessitating efficient indexing and optimized function design to minimize execution time and cost. Furthermore, managing the asynchronous nature of real-time database listeners was a crucial skill developed during the frontend implementation.

FUTURE ENHANCEMENTS

The HerHealth application has several exciting possibilities for future development to increase its complexity and utility. Firstly, implementing integration with popular wearable devices (like smartwatches or fitness rings) would automate the logging of Heart Rate and activity data, significantly enhancing user convenience and data accuracy. Secondly, the current personalization is limited to diet and exercise; future enhancements could include a Mood and Stress Index module that offers guided meditation or stress-reduction techniques dynamically based on logged symptoms and cycle phase. Finally, evolving the cycle prediction algorithm from a weighted moving average to a more sophisticated Machine Learning model, such as a Recurrent Neural Network (RNN), would allow for even more precise and adaptive predictions for highly irregular cycles, improving the core value proposition of the tracker.

CONCLUSION

The HealthMonitorApp successfully implements a fundamental command-line utility for tracking basic health vitals and menstrual cycle start dates.

This project demonstrates core object-oriented programming (OOP) principles in Python, utilizing a class structure to encapsulate data (vitals_history, cycle_history) and methods (log_period, add_health_metric, view_vitals, view_cycle_history) for data handling. The application provides a simple, menu-driven interface and includes basic error handling in the log_period method to ensure valid date input.

In summary, the application establishes a solid, functional foundation for a personal health tracker. The next logical steps for enhancing this project would involve adding data persistence (saving and loading data using files) and implementing advanced analytics (such as calculating cycle averages or predictions) to significantly increase its utility.

DONE BY

KRIPA S KANNAN

REG NO:25BOE10151

BRANCH:BTECH BIOENGINEERING