Risk Analysis

Risk Event	**Likelihood (L)**	**Impact (I)**	**Risk Level**
Data Accuracy and Quality (1)	Low	High	High
Model Training and Validation (2)	Moderate	High	Moderate
Bias and Fairness (3)	Moderate	High	Moderate
Interpretability/Customer Perception (4)	Low	Moderate	Low
Security and Privacy (5)	Moderate	High	High
User Feedback and Improvement (6)	Moderate	High	Moderate
User Education and Engagement (7)	Moderate	Moderate	Moderate
Continuous Monitoring (8)	Moderate	Moderate	Moderate
Server Maintenance (9)	High	High	High
Latency Handling (10)	Moderate	High	High
Regulatory Obligations (11)	Low	High	Moderate
Testability (12)	Moderate	Moderate	Moderate

Quality and accuracy of data:

Risk: Predictions may not be correct if the data used (BMI, cholesterol, etc.) is wrong or of poor quality.
Fix: Add checks to make sure the data is correct, and teach people how to give correct information. To improve accuracy, keep adding the most recent medical study to the model on a regular basis.

Validation and training of models:

Risk: If the model is too well or too poorly fitted during training, it could affect how well it does on real-world data.
Fix: For training and validation, use datasets that are varied and representative. Use cross-validation methods and keep an eye on and update the model all the time to keep up with changes in health trends.

Fairness and bias:

Risk: If the training data is biased, the model might make wrong predictions, and there's a chance that unfair results will happen, especially if the dataset doesn't show the whole community.
Fix: Check the model for bias on a regular basis. Use methods like fairness-aware machine learning and make sure that both the development team and the training data have a lot of different kinds of people.

Ability to interpret:

Risk: Users and healthcare experts may not trust the model because it is hard to understand.
Fix: Make sure that your models are clear and that you explain your predictions well. Use machine learning techniques that can be interpreted or ways that can be interpreted after the fact.

Safety and Privacy:

Risk: There is a chance that someone will get to private health data without permission, which is called a data breach.
Fix: User’s data is in the local device only.

Educating and engaging users:

Risk: People may misunderstand the app's predictions or put too much faith in them.
Fix: Make it clear how to understand the data. Stress that the app is only a supplement and not a replacement for medical help from a professional. Encourage people to talk to healthcare workers for more information.

Monitoring and updates all the time:

Risk: Without regular updates, the model may become outdated and less accurate over time.
Fix: Establish a system for continuous monitoring of model performance and update the app regularly based on emerging medical research and new data.

User Feedback and Making Things Better:

Risk: Ignoring user feedback may lead to dissatisfaction and reduced effectiveness of the app.
Fix: Ask users for comments and use it to make the app better. Keep lines of communication open so that you can quickly answer user concerns.

Server Maintenance:

Risk: The server must be able to handle large loads without high latency.
Fix: Periodic check of the server and cleaning up the cache memory to ensure quick responses.

Latency Handling:

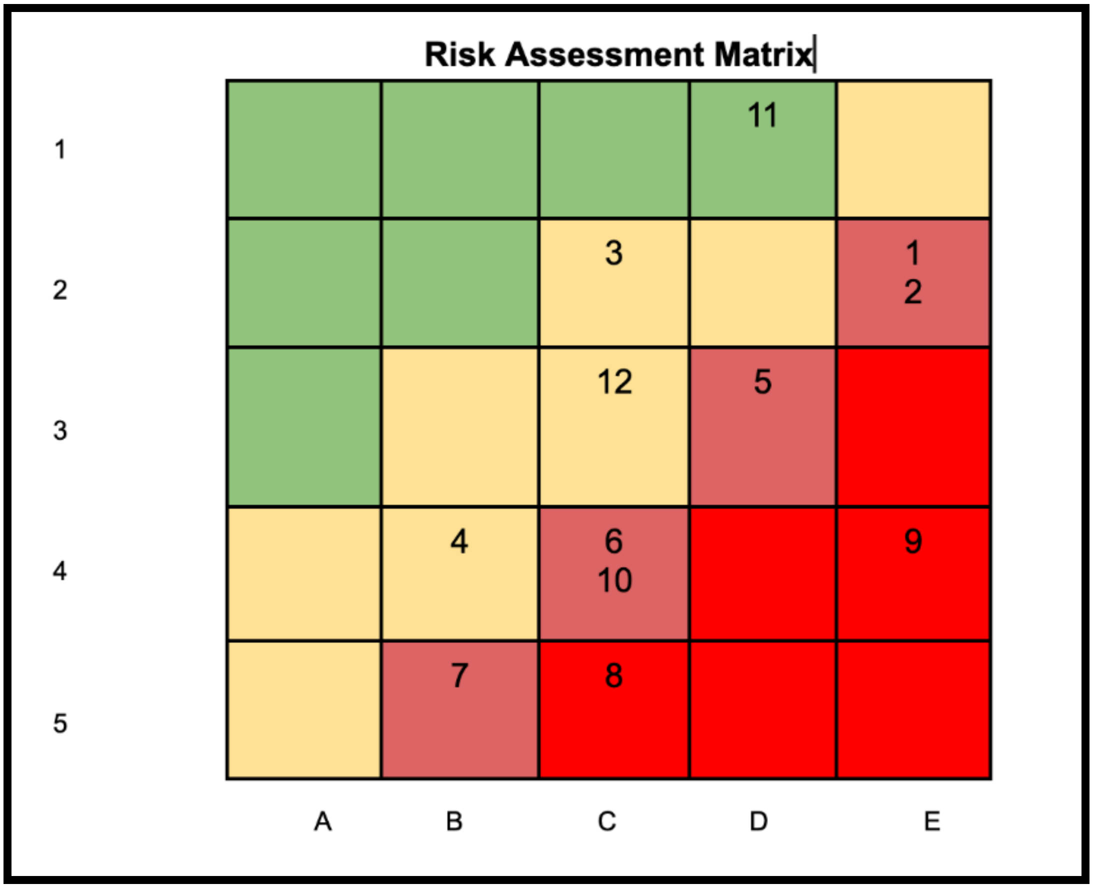
Risk: Latency in operations must be minimal.
Fix: Can have multiple distributed servers with more robust exception handling mechanisms.

Regulatory Obligations:

Risk: Failure to comply with healthcare regulations and data protection laws may result in legal penalties and damage to reputation.
Fix: Implement strict data protection measures, conduct regular audits to ensure compliance, and stay updated with relevant regulations through legal counsel or compliance experts.

Testability:

Risk: Inadequate testability may lead to difficulties in identifying and resolving software defects, impacting the app's reliability and user experience.
Fix: Develop comprehensive test plans covering unit testing, integration testing, and user acceptance testing to ensure all aspects of the app are thoroughly tested. Implement automated testing where possible to streamline the testing process and facilitate continuous integration and deployment.



Risk Assessment Matrix

	A	B	C	D	E
1	🟩	🟩	🟩	🟩 11	🟨
2	🟩	🟩	🟨 3	🟨	🟧 1 2
3	🟩	🟨	🟨 12	🟧 5	🟥
4	🟨	🟨 4	🟧 6 10	🟥	🟥 9
5	🟨	🟧 7	🟥 8	🟥	🟥

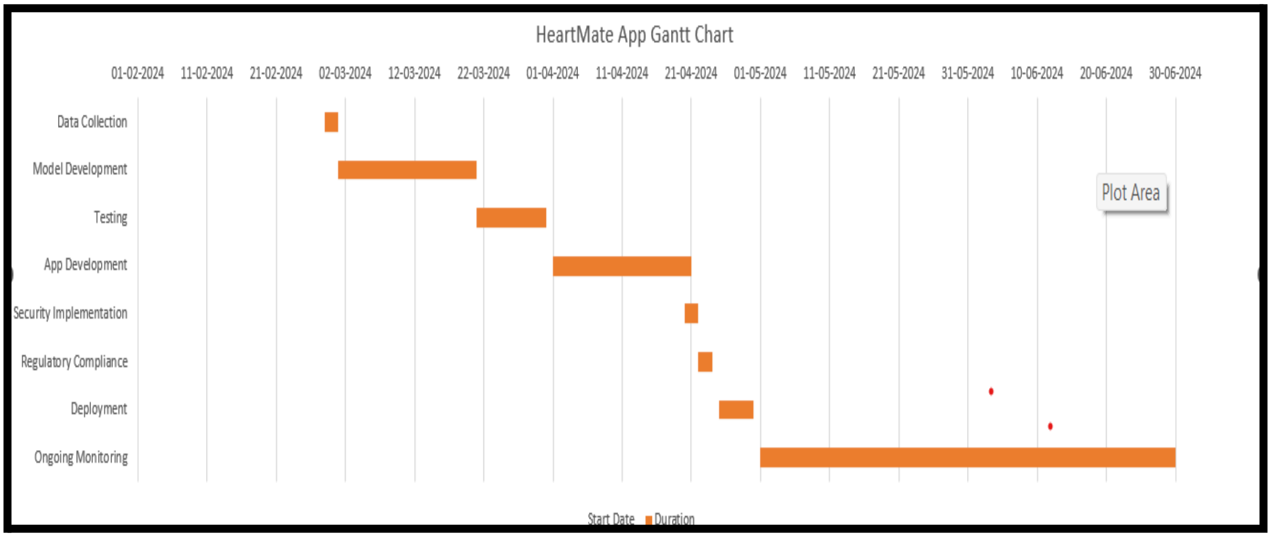
Legends:

Likelihoods	Consequences
1- Unlikely	A - Insignificant
2- Seldom	B - Marginal
3- Occasional	C - Moderate
4- Likely	D - Critical
5- Definite	E - Catastrophic

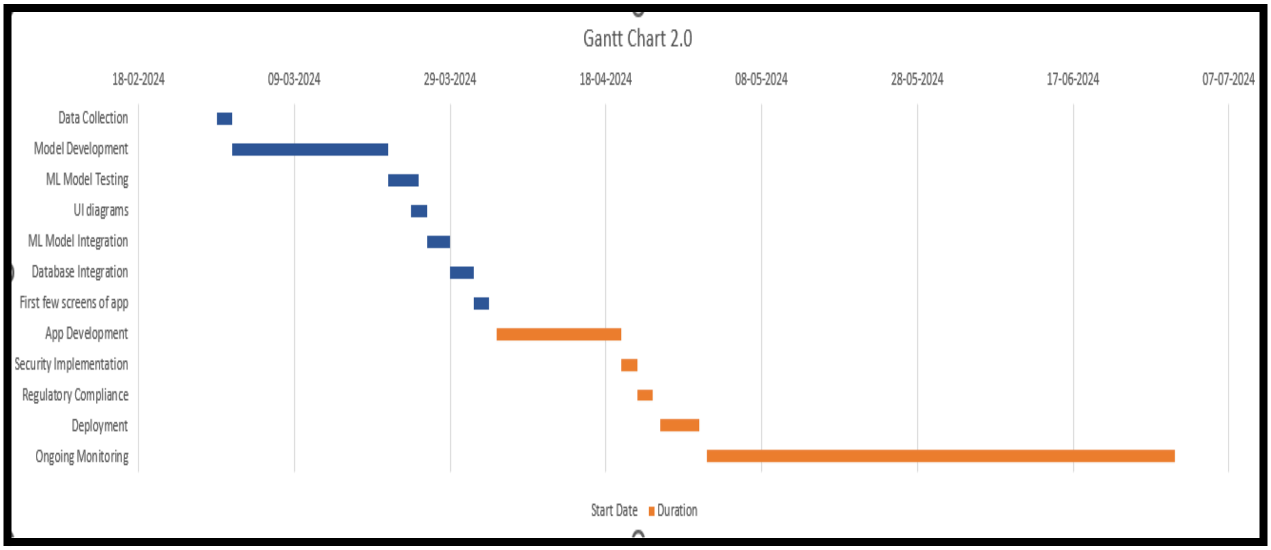
Color Legends:

🟩 Low risk - No Further Action	🟨 Medium risk - Can take action	🟧 High Risk - Further Action Necessary	🟥 Extreme Risk- Act Now

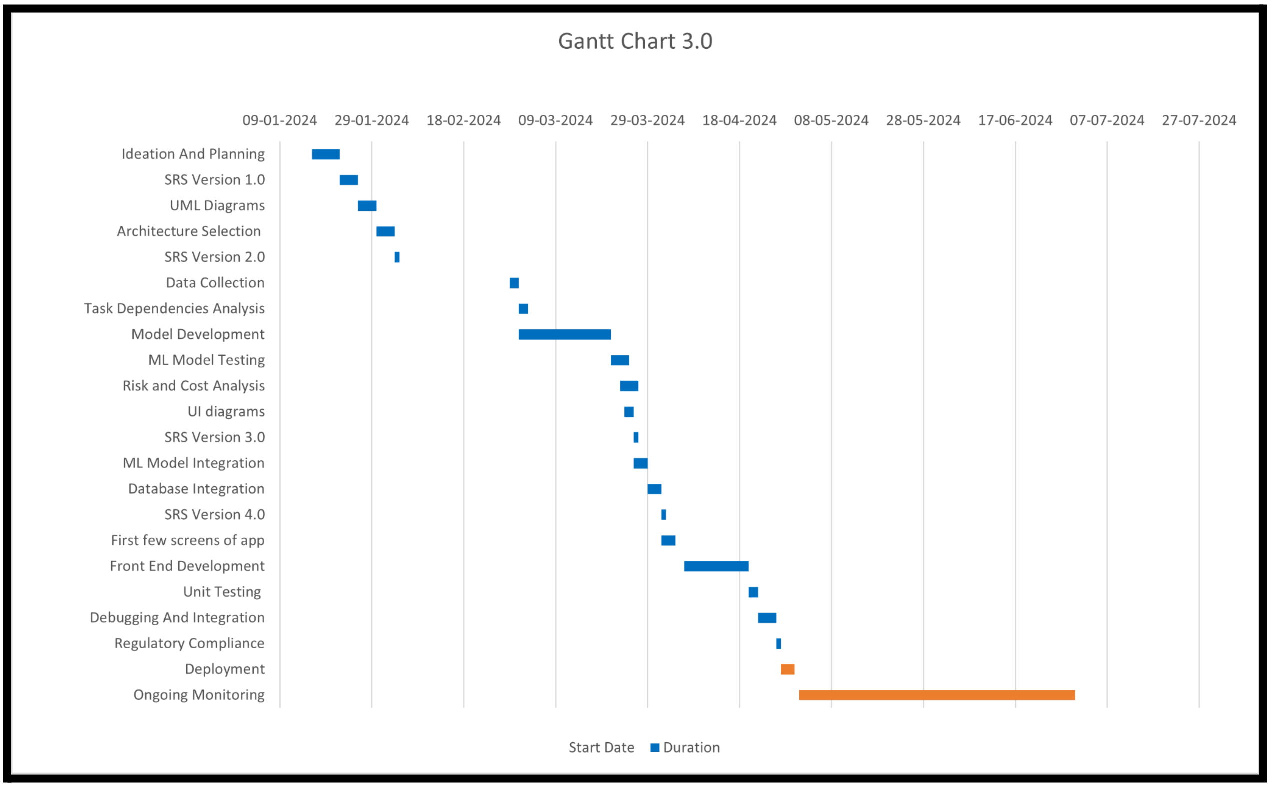
Tentative Gantt Chart



Fractal-II Gantt Chart



Final Gantt Chart



Cost Analysis Report

Development Costs:
Personnels (Developers, Data Scientists, Project Manager): This category usually takes up a big chunk of the budget because it requires skilled workers to build and create the machine learning models, app architecture, and overall development.

Testing Costs:
Quality Assurance (Testing Team): Thorough testing is needed to make sure that the predictions are correct and reliable, so this takes up a big chunk of the cash.
Testing Tools and Environments: The total cost of testing goes up when you buy tools and environments for thorough testing.

Deployment Costs:
App Development: It costs a lot to make an app that works well and is easy for people to use, especially if it has features like user interfaces, data visualization, and connecting to health tracking devices.
Implementing Security: Strong security steps are needed to keep private health information safe, which can cost a lot.
Regulatory Compliance: Making sure that healthcare laws are followed may require talking to a lawyer and making changes to the app, which adds to the cost.

Maintenance Costs:
Continuous Monitoring Tools: For long-term success, it's important to buy tools that can keep an eye on the app's speed, data accuracy, and any problems that might come up.
Staff (Maintenance Team): For responding to user comments, fixing bugs, and making sure the app keeps working well, it needs ongoing maintenance and support from a dedicated team.

Miscellaneous Costs:
Legal and Compliance: In healthcare and data security, following the law and moral standards may require legal advice and compliance measures.
Contingency Fund: Setting aside some money for unplanned costs gives you the freedom to deal with problems or changes in the project's scope that come up out of the blue.

Other points to consider:

Costs of Personnel: Skilled workers in development, data science, and compliance add a lot to the total budget.
Regulatory Compliance and Security: Making sure that healthcare laws are followed and that strong security measures are put in place can take a lot of time and money, but it is necessary to protect user trust and data.
Testing and Quality Assurance: The app's plans need to be tested thoroughly in order to be accurate and reliable, so this takes up a big chunk of the budget.
Continuous Tracking and Maintenance: Tools for ongoing tracking and maintenance teams help the app work well and be successful in the long run.

Task Dependencies

Data Collection: Predecessor task for all subsequent tasks
ML Model Development: FS wrt data collection (successor task) and testing (predecessor task)
ML Model Testing: FS wrt model development (successor task), SS with respect to app development
UI Diagrams: FS wrt app development (predecessor task)
ML Model Integration: FS wrt ML model testing (successor task), FS wrt app development (predecessor task)
Database Integration: FS wrt app development (predecessor task)
App Development: FS wrt model development (successor task), FS wrt UI diagrams designing (successor task), FS wrt ML model integration (successor task), FS wrt database integration (successor task), FF wrt unit testing (successor task), FF wrt debugging and integration (successor task), FF wrt regulatory obligations (successor task)
Unit Testing: FF wrt app development (predecessor task)
Debugging and Integration: FF wrt app development (predecessor task)
Regulatory Obligations: FS wrt security implementation (successor task), FF wrt to deployment (predecessor task), FF wrt app development (predecessor task)
Deployment: Successor task for all aforementioned tasks, FF wrt to ongoing monitoring (predecessor task)
Ongoing Monitoring: Successor task of all aforementioned tasks

Critical Path:

Data Collection → Model Development → Model Testing→ UI Diagrams → Model Integration → Database Integration →Obligations → Deployment → Ongoing Monitoring