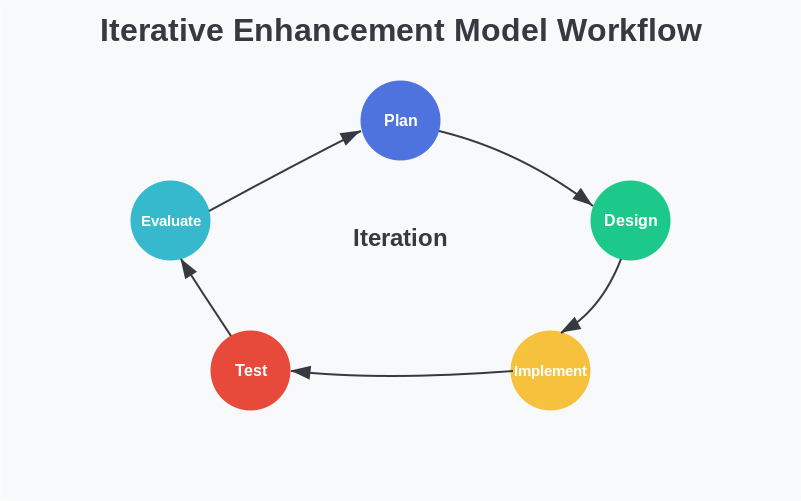
**Research Methodology.**

The Iterative Enhancement Model is a software development methodology that focuses on building a system through repeated cycles (iterations) and in smaller portions at a time. This approach allows developers to take advantage of lessons learned during the development of earlier parts or versions of the system. Each iteration follows a complete software development cycle**;**

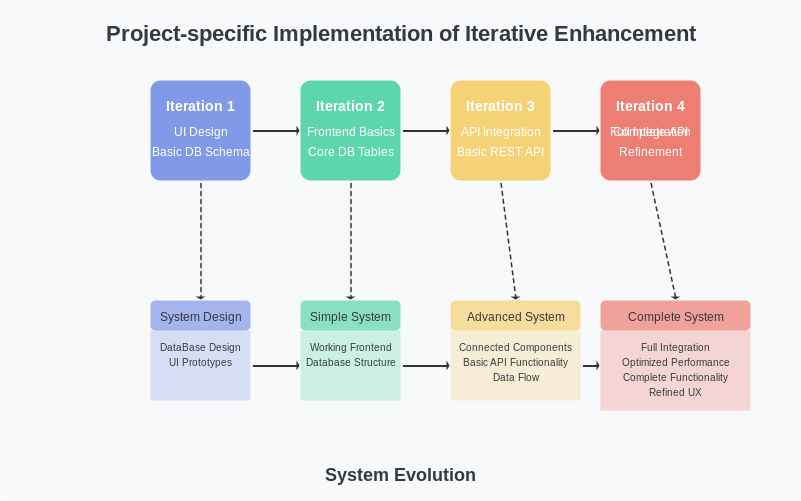


The product is defined, implemented, and tested incrementally, with each iteration producing a working version that is more complete than the previous one. Unlike the traditional Waterfall approach, this model does not require complete specifications upfront, allowing for greater flexibility and adaptation as the project evolves. According to Basili and Turner (1975).

### Major Iterations for the project.

The project i split into 4 major iterations which will deliver the project from a concept to a working system. Each iteration will have its own minor iterations to achieve it’s end goals. Each iteration consists of planning, design, implementation, testing, and evaluation before moving to the next phase. These iterations include the;

1. **Designs and wireframe:** Database and frontend designs and ideas.
2. **Simple System:** Frontend prototype, database schema and API documentation.
3. **Advanced system:** Begin Rest API construction and integration using API documentation.
4. **Complete System:** Finalized Rest API and integration. System testing and deployement.



## Justification for Model Selection

### 1. Team Adaptability

This model is ideal for a team with varying experience levels. By starting with basic functionality and progressively adding more complex modules, team members can learn incrementally and build confidence.

### 2. Technology Integration

Given that our project involves multiple distinct technologies (frontend design, database, REST API development), an iterative approach allows us to develop and test each component separately before integrating them.

### 3. Feedback

By delivering working software increments at the end of each iteration, we can evaluate actual functionality instead of abstract designs, leading to better alignment with software needs and higher satisfaction.

### 4. Risk Mitigation

Detecting and addressing issues early prevents technical debt and major project setbacks.

## Technologies of choice

## 1. Design

**Selected Technology:** Figma

## 2. Database

**Selected Technology:** PostgreSQL

## 3. Frontend

**Selected Technologies:** React, TanStack Router, Vite, Tailwind CSS, shadcn components library.

## 4. REST API

**Selected Technology:** Java Spring Boot

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |