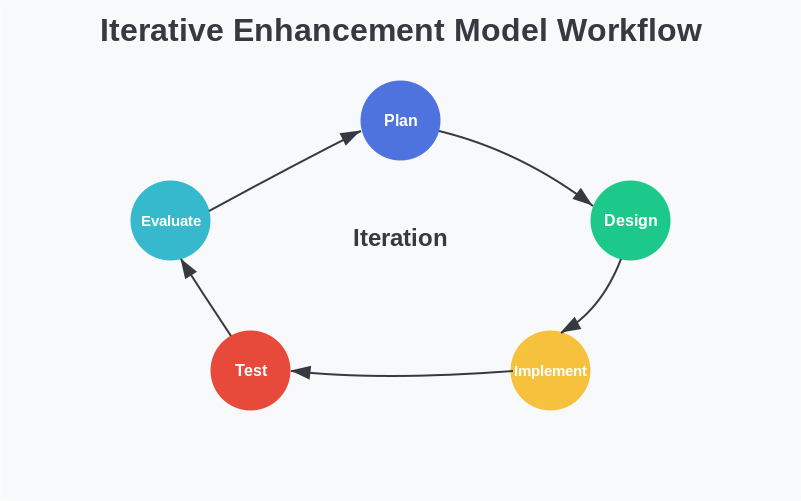
**Research Methodology.**

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).

## Justification for Model Selection

### 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.

### 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.

### 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.

### Risk Mitigation

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

### Techenical Approach

## Technologies of choice

## 1. Design****:**** Figma

## 2. Database****:**** PostgreSQL

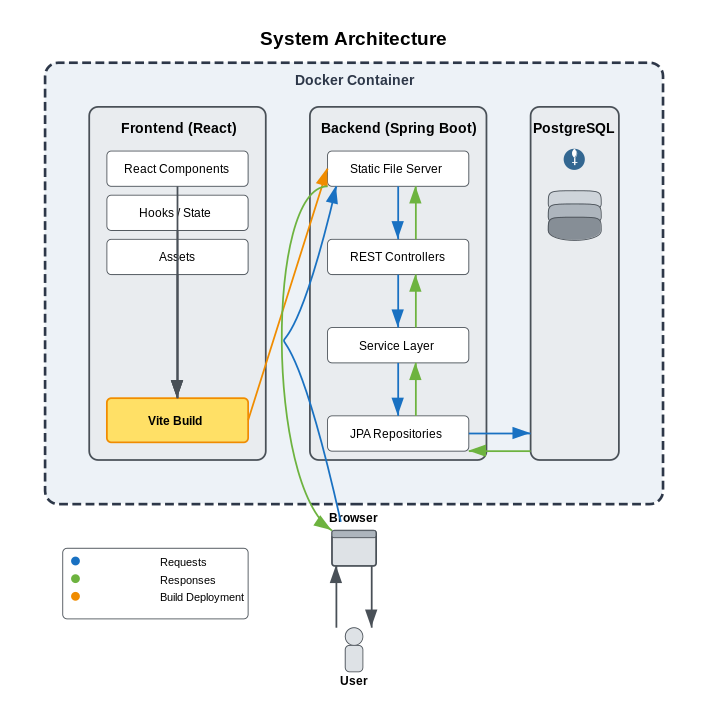
## 3. Frontend****:**** React, TanStack Router, Vite, Tailwind CSS, shadcn components library.

## 4. REST API****:**** Java Spring Boot

**5. Containerization:** Docker.

**System Architecture.**

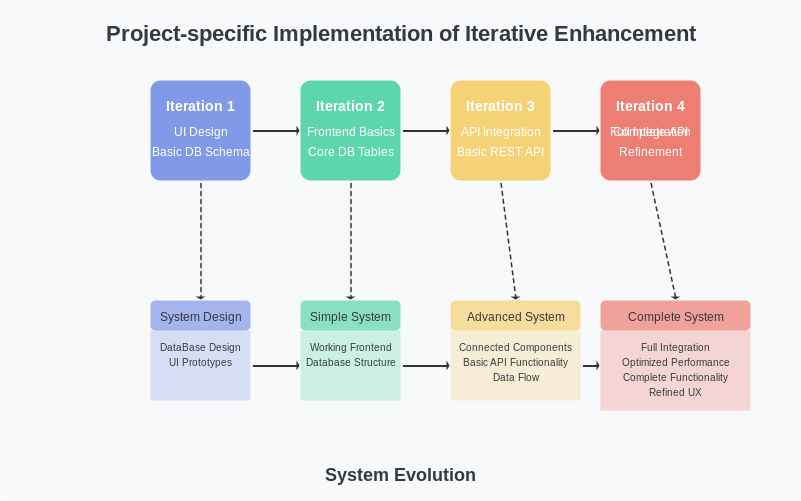
The system uses a modular architecture that separates the frontend and database by means of a Rest API. To run a single application we will package the frontend code into raw html, javascript and css using vite, and then render these files statically through the backend application. The system is to be containerised using docker.



### 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.



### ****Testing****

We will implement a comprehensive testing strategy to ensure the reliability, functionality, and performance of the system. Our testing approach will include:

1. **Unit Testing** – Each component will undergo rigorous unit testing using automated test suites to validate individual functionalities and catch errors early in development.
2. **Integration Testing** – After each development iteration, we will conduct integration testing to ensure seamless interaction between various components and modules.
3. **System Testing** – Once all iterations are complete, we will perform full system testing to validate the end-to-end functionality, security, scalability, and performance of the application.
4. **Continuous Testing & Monitoring** – Automated tests will be integrated into our CI/CD pipeline to detect regressions early, and real-time monitoring will be set up post-deployment to track system stability.

### ****Project Management****

To ensure efficient development and collaboration, we will employ the following project management strategies:

* **Version Control & Collaboration** – We will use Git and GitHub for code management, progressive feature development, and collaborative contributions. Branching strategies such as feature branches and pull requests will be enforced to maintain code quality.
* **Regular Stand-up Meetings** – Short daily or weekly stand-up meetings will be held to review progress, address challenges, and ensure alignment among team members.
* **Code Reviews & Quality Assurance** – Peer code reviews will be conducted before merging changes into the main branch to maintain code integrity and enforce best practices.
* **Issue Tracking & Task Management** – Tools like GitHub Issues or project boards will be used to track bugs, enhancements, and development progress.

### ****Documentation and Training****

To facilitate smooth adoption, maintenance, and scalability, we will create and maintain comprehensive documentation:

* **Technical Documentation** – Includes API documentation, database schema, system architecture, and deployment guides. This will be continuously updated as the project evolves.
* **User Guides & Manuals** – A structured guide will be provided for end-users, covering system usage, features, and troubleshooting steps.
* **Developer Onboarding & Training** – Detailed setup instructions, coding standards, and best practices will be documented to streamline onboarding for new contributors.
* **Knowledge Transfer Sessions** – Before project handoff, training sessions will be conducted for stakeholders and support teams to ensure smooth operation and maintenance of the system.

### ****Deployment****

The deployment strategy will be designed for scalability, security, and maintainability:

* **Staging & Production Environments** – A staging environment will be used for testing before production deployment to minimize risks.
* **Hosting & Infrastructure** – The application will be deployed on a reliable cloud platform (e.g., AWS, Vercel, or Firebase) based on project needs.
* **Post-Deployment Monitoring** – Logging, error tracking, and performance monitoring tools will be set up to detect and resolve issues proactively.
* **Backup & Disaster Recovery** – Regular backups and a rollback strategy will be in place to prevent data loss and system failures.

**Project Timeline.**

**Gantt Chart.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Duration(**weeks**)** | | | | | | | | | | | | | | | | | | | | | |
| Febuary | | | | March | | | | April | | | | May | | | |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 12 |  |  |  | 13 |  |
| First iteration |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Second iteration |  |  | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Third iteration |  |  |  |  |  |  |  | | | | | | | |  |  |  |  |  |  |  |  |
| Forth iteration |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | | | | |  |  |
| System Testing and deployment |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| documentation |  | | | | | | | | | | | | | | | | | | | | | |