Assignment - 2

Software Development Life Cycle Model Connect (Spiral Model)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, GUWAHATI

Anant Sharma (2101036)

Aman Yadav (2101032)

Adarsh Agrawal (2101015)

Contents:

1. Determine Objectives, Alternatives, and Constraints:

Objectives:

- Develop a web-based platform connecting pet owners with rehabilitation centers.
- Streamline the process of booking rehabilitation services.
- Ensure effective communication between users and rehabilitation centers.
- Provide a secure and user-friendly experience.

Alternatives:

- Explore different technologies for the front-end (ReactJS and Next.js).
- Consider various database options (MongoDB).

Constraints:

- Dependence on external services like payment gateways.
- Need for a stable internet connection for users.
- Availability of a rehabilitation centre nearby.

2. Identify and Resolve Risks:

2.1) Reliability Issues with External Payment Gateways:

Potential Risks:

- Dependency on third-party payment gateways may introduce potential points of failure.
- Unplanned downtime or technical issues with the payment gateway could disrupt transaction processes, impacting user trust.

Strategies to Mitigate:

- Implement a fallback mechanism that allows the system to switch to an alternative payment gateway in case of issues with the primary one.
- Regularly monitor the performance and reliability of the chosen payment gateways.
- Provide clear communication to users about the payment process, potential delays, and alternative payment methods in case of issues.

2.2) Data Security Concerns:

Potential Risks:

- Risks associated with the secure storage and transmission of user and animal data, especially during online transactions.
- Possibility of unauthorized access or data breaches compromising sensitive information.

Strategies to Mitigate:

- Implement end-to-end encryption for all user interactions with the platform, ensuring that sensitive data is secure during transmission.
- Regularly conduct security audits to identify and address potential vulnerabilities.
- Employ strong authentication mechanisms, including secure session management, to protect user accounts and data.

2.3) Technology Compatibility Issues:

Potential Risks:

- Incompatibility issues with different web browsers and devices may result in an inconsistent user experience.
- Challenges in integrating external services and APIs due to changes or updates in their specifications.

Strategies to Mitigate:

- Perform thorough compatibility testing across major web browsers (Chrome, Firefox, Safari) and devices (desktops, laptops, tablets, smartphones).
- Keep abreast of updates in external services and APIs, ensuring timely adjustments to maintain compatibility.

3) Development Planning:

3.1) Model View Controllers (MVC) Model:

Introduction to MVC Model:

The development approach for the Animal Rehabilitation Website will adhere to the Model View Controllers (MVC) model.

MVC is a software architectural pattern that divides an application into three interconnected components: Model, View, and Controller.

Components of MVC in the Animal Rehabilitation Website:

Model (Data and Business Logic):

Represents the data and business logic of the application. In the context of the website, this includes the structures and operations related to user profiles, pet data, rehabilitation center details, and other essential information.

View (User Interface):

Responsible for presenting information to users and receiving their input.

In the website, views encompass the frontend user interfaces (UI) that pet owners, rehabilitation centers, and administrators interact with. The UML diagram will be the methodology used to arrive at the end design.

Controller (Application Logic):

Acts as an intermediary between the Model and the View, handling user input and updating the Model accordingly.
In the Animal Rehabilitation Website, controllers manage the logic that connects the frontend UI (Views) with the backend database (Model).

3.2) User Module:

Frontend UI (Views):

- Refers to the visual components that users interact with when accessing the website.
- For the User Module, this includes the registration form, search functionality, profile management, and the ability to leave reviews and ratings.
- The UML diagram will be the methodology used to arrive at the end design.

Database with Dummy Data:

- In the development phase, a database populated with dummy data will be employed.
- Dummy data allows developers to simulate real-world scenarios during testing and development, ensuring the system functions as intended before actual data is utilized.

Controllers (Logic connecting UI with Database):

- Controllers for the User Module handle the logic that facilitates communication between the frontend UI and the backend database.
- These controllers manage user registration, search functionalities, profile editing, and interactions with reviews and ratings.

3.3) Rehabilitation Centre Module:

Frontend UI (Views):

- Similar to the User Module, the Rehabilitation Centre Module's frontend UI consists of interfaces for registration, profile management, and interaction with animals under their care.
- The UML diagram will be the methodology used to arrive at the end design.

Database with Dummy Data:

- A database with dummy data will be utilized for the Rehabilitation Centre Module during development.
- This allows for the testing of functionalities related to managing animal information, updating services, and handling doctor details.

Controllers (Logic connecting UI with Database):

- Controllers specific to the Rehabilitation Centre Module manage the logic connecting the frontend UI to the backend database.
- These controllers handle tasks such as registration, profile management, updating animal information, managing doctor details, and processing booking-related transactions.

4) Testing:

4.1) Phase 1:

4.1.1) Database Testing:

Objective:

- Evaluate the system's robustness in handling various data scenarios.

Approach:

- Populate the database with error-prone data, including edge cases, invalid inputs, and extreme values.
- Verify that the system can manage and process this data without compromising functionality or security.

Benefits:

- Identifying potential vulnerabilities or weaknesses in handling diverse data sets.
- Ensuring data integrity and security measures in the face of unexpected inputs.

4.1.2) Performance Testing:

Objective:

- Assess the system's performance under high-load conditions.

Approach:

- Overload the system by simulating a large number of concurrent user requests.
- Monitor response times, system resource utilization, and identify potential bottlenecks.

Benefits:

- Determining the system's scalability and ability to handle increased user loads.
- Uncovering any performance-related issues, such as slow response times or system crashes.

4.1.3) Compatibility Testing:

Objective:

- Verify the application's compatibility with a range of devices.

Approach:

- Test the application on older and subpar devices with varying screen sizes and resolutions.
- Evaluate the responsiveness and functionality across different browsers and devices.

Benefits:

- Ensuring a consistent and user-friendly experience across a diverse range of devices.
- Identifying and addressing compatibility issues early in the testing phase.

4.2) Phase 2:

4.2.1) Beta Testing:

Objective:

- Gather user feedback and identify potential issues in a real-world environment.

Approach:

- Release the application to a group of beta testers, including actual users and stakeholders.
- Encourage testers to use the application as they normally would, reporting any issues or providing feedback.

Benefits:

- Collecting valuable insights from real users regarding usability, bugs, and additional features.
- Addressing user concerns and making necessary adjustments before the full-scale release.

4.2.2) User Acceptance Testing (UAT):

Objective:

- Ensure the application meets user expectations and business requirements.

Approach:

- Collaborate with beta testers to perform UAT on specific features and functionalities.
- Evaluate the overall user experience and assess whether the application aligns with the project's objectives.

Benefits:

- Confirming that the application meets user needs and functions as intended.
- Building confidence in the reliability and usability of the application before the official launch.

5) Planning for Next Iteration:

Plan the next iteration based on feedback and identified areas for improvement.

6) Justification:

The Spiral Lifecycle Model is well-suited for our project due to the following reasons:

- **Iterative Development:** The nature of the project involves developing a web platform with complex functionalities. The iterative approach allows gradual enhancements and refinement based on continuous feedback.

- **Risk Management:** The project has inherent risks such as dependence on external services and the assumption of stable internet connections. The Spiral Model's focus on risk analysis and management aligns well with the project's characteristics.

- **Adaptability:** The project scope includes features like user authentication, payment integration, and secure communication. The Spiral Model's adaptability allows for adjustments based on evolving requirements and technological advancements.

- **Continuous Evaluation:** Regular evaluation and feedback collection are crucial for a project involving multiple stakeholders. The Spiral Model's iterative cycles provide opportunities for continuous assessment and refinement.