Project (Phase 2)

Student: Albion Osmanaj

1. Chosen Development Model:

Agile: In the case of this project seeing as I am a one person team this development model fits the criteria perfectly through the implementation of different methods.

Develop your project iteratively by dividing it up into smaller, more doable tasks or user stories. Work on these assignments in an iterative manner, making sure that each iteration adds more value.

Prioritization: Keep a backlog of features or tasks and order them according to their worth and significance. This will assist you in maintaining concentration on the most important aspects of the success of your project.

Timeboxing: Establish regular time slots for yourself, such once a week or twice a week, to assess your progress, plan out the next steps, and reorganize your priorities. This establishes a cadence for your own growth and guarantees that you consistently evaluate your advancement and make necessary adjustments.

Continuous Improvement: Take time to reflect on your work regularly. Review what went well, what didn't, and what you can improve for the next iteration. This might involve looking at your code quality, your development process, or your project management practices.

2. User Requirements:

Stakeholders and User Stories:

Our stakeholders would be people such as the users who are looking for their services to be fulfilled, like babysitting, plumbing , cleaning etc.

The service providers themselves like babysitters, cleaners, plumbers, etc. Basically everyone who is able to work and is willing to use our app for their freelancing purposes.

Investors and shareholders that have a financial stake in the success of the company. That are interested in the company's profitability, growth prospects, and overall market performance.

Companies providing technology solutions such as cloud infrastructure providers, payment processors, data analytics firms, and software developers, are also stakeholders. They contribute to the platform's functionality, reliability, and scalability.

3. Functional Requirements:

Brief Description and Criteria:

- My app should provide a platform where people can both find work and find the help they need.
- It needs to be able to let people of all professions that apply to open a profile from which they can state their profession, skills and other useful criteria and let them work as a freelancing or even a private business with the help of the app.
- In contrary it should let people open a profile and start hiring service providers who can solve their problems.

3. Functional Requirements:

Brief Description and Criteria:

- My app should provide a platform where people can both find work and find the help they need.
- It needs to be able to let people of all professions that apply to open a profile from which they can state their profession, skills and other useful criteria and let them work as a freelancing or even a private business with the help of the app.
- In contrary it should let people open a profile and start hiring service providers who can solve their problems.
- Implement a map and a review system to be able to have feedback in-between users and service providers.

4. Non-Functional Requirements:

- The app should be fast, precise and dependable.
- It should be a smooth experience with no errors.
- All the functions like the map feature and the review system should be accurate and up to date.
- Every feature should work as intended with no bugs.

5. Application Specifications:

Front-end Components:

1. User Interface (UI): The UI of the app, that the users will face, it will be simple and minimalistic so it can be easy to use for every person of any age.
2. Client-side Frameworks: Front-end frameworks such as React.js or Angular.js may be used to build dynamic and responsive user interfaces. These frameworks enable the development of interactive components and facilitate data binding between the UI and back-end services.
3. Mobile Applications and a WEB page to provide seamless use between both services.

Back-end Components:

1. Server-side Application: The server-side application handles core business logic, data processing, and communication with external services. It manages user authentication, listing management, booking workflows, messaging, and other essential functionalities.

2. Database: A relational database management system (RDBMS) such as PostgreSQL or MySQL will be used to store data related to users, reviews, and other application entities. The database ensures data integrity, consistency, and efficient retrieval.

3. API Layer: An application programming interface (API) layer exposes endpoints for client-side applications to interact with back-end services. This includes RESTful APIs or GraphQL endpoints for performing CRUD operations on resources like users, listings, bookings, and reviews.

4. Authentication and Authorization: The application implements authentication mechanisms such as OAuth or JSON Web Tokens (JWT) to secure access to protected resources. Authorization rules are enforced to ensure that users can only access data and perform actions they are authorized for.

- **External Integrations:**

1. Payment Gateways: Integration with third-party payment gateways such as Stripe, PayPal, or Braintree allows users to make secure payments for bookings. The application handles payment processing, fee calculations, and transaction management.
2. Geolocation Services: Integration with geolocation services like Google Maps or Mapbox enables features such as location-based search, map visualization of listings, and proximity-based recommendations.
3. Messaging Services: Integration with messaging services such as Twilio or SendGrid facilitates real-time communication between users and service providers including hiring inquiries, notifications, and support messages.
4. Analytics and Monitoring: Integration with analytics platforms like Google Analytics or Mixpanel provides insights into user behavior, usage patterns, and performance metrics. Monitoring tools such as New Relic or Sentry help track errors, performance issues, and system health.

Most of these features will be implemented at further stages of the app lifecycle, depending on its success.

- **User Interface**