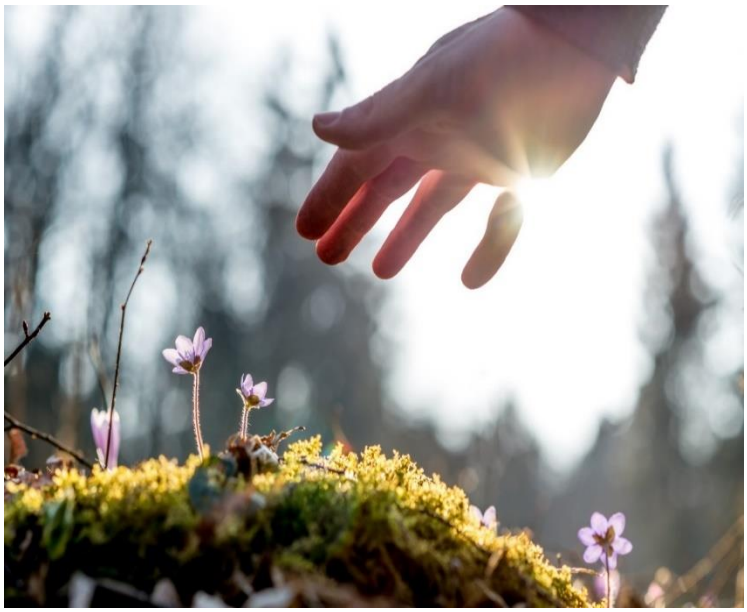

Proposal to the Office of Systems Integration (OSI)

Request for Information (RFI) #75001

For Agile Development Pre-Qualified (ADPQ) Venfor Pool

Architecture Review

Jun 9th, 2016

Strategy! Innovation! Transformstion!

Revision History

Version	Date	Description of Updates	Author

Table of Contents

List of Tables

NO TABLE OF FIGURES ENTRIES FOUND.

LIST OF FIGURES

1 Introduction

The purpose of this document is to provide a overview of the architecture, design and implementation approach, best practices used for the SafeKids application. This document provides high level description of the logical and physical architecture, data model and the business and technical services used to implement the application.

2 Logical Architecture

The logical architecture for the SafeKids application is shown in Figure 2-1 and is described below.

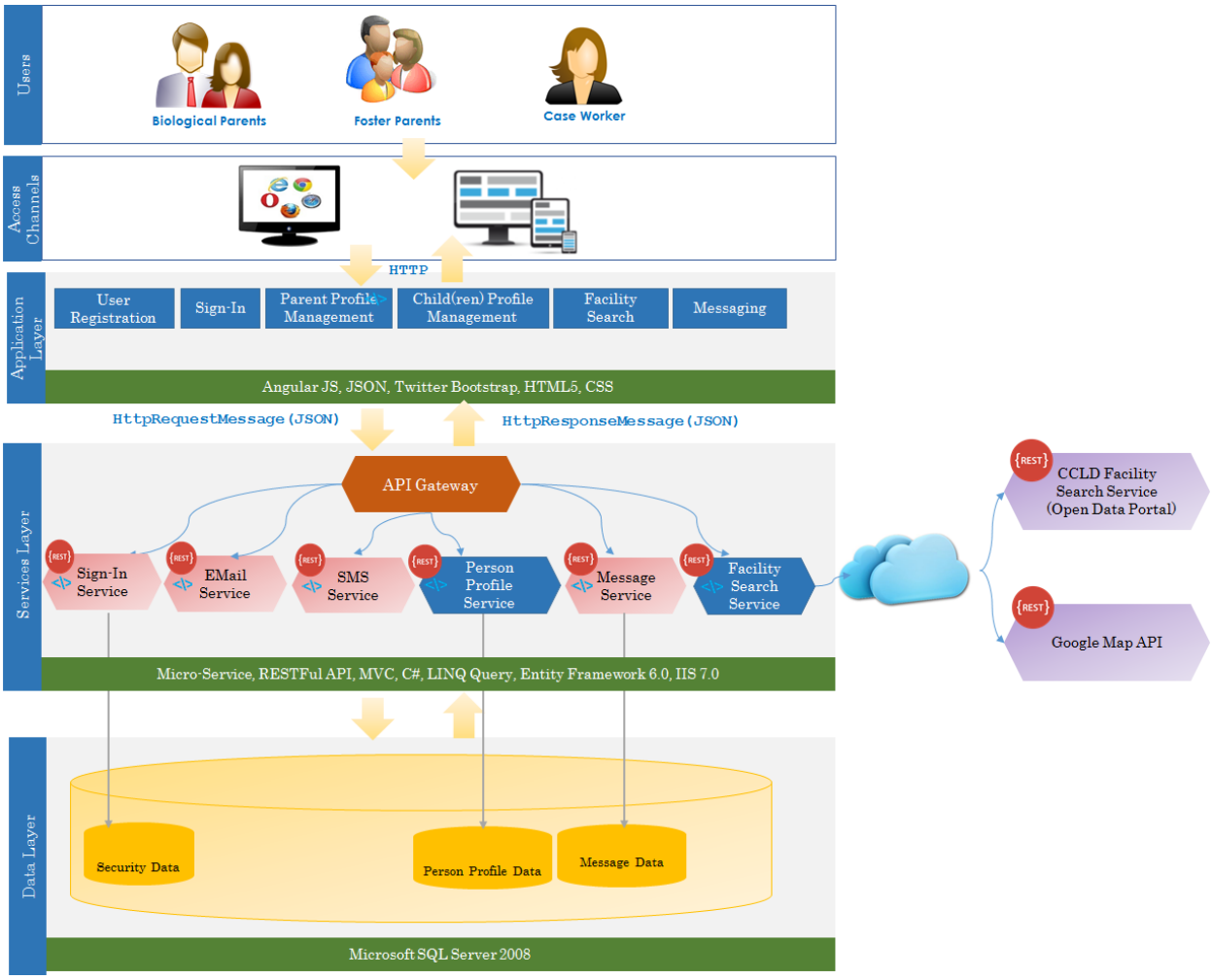

Figure 2-1 Logical Architecture Diagram

2.1 Architecture Approach and Best Practices

The SafeKids application architecture is implemented using the following architectural features/best practices:

1. **N-Tier Distributed Architecture** – supports horizontal and vertical scalability

2. **Technology Agonstic Application Layer** – The application is implemented using tools/technologies that are completely technology platform (such as Java/JEE, MS .NET etc) agonstic.

3. **Responsive UX/UI Design** – The web based User Interface (UI) uses Responsive Design techniques/tools.

a. The Application is accessible through any web browser such as *Google Chrome, Mozzila FireFox, MS Internet Explorer, Safari and Opera* etc.
b. Accessible through wide range of Mobile devices (tablets/cell phones) running on *iOS, Android, and Microsoft* operating systems.

4. **Section 508 Compliance –** The user interface is compliant to the Section 508 requirements.

5. **Microservice Based Architecture**
   a. The Business and Technical capabilities requirements are derived from the user stories
   b. The Business and Technical Services are implemend using Microservices API architecture as RESTful web services.
   c. **Autonomous Services:** Each Microservice is designed to have complete authority to perform CRUD (Create, Read, Update, Delete) operations on the specific set of domain data. No other services can perform CRUD operation on this domain data set. This makes the Microservices *Autonomous API*.
   d. The Microservices can be designed, implemented, built and deployed independently

6. **Reusable Technical Services** – The Microservices API design approach carefully separates Business Services from Technical Services. The Technical Servics are business context agonstic and can be reused by any other business process/services. The Technical Services APIs such as ***Sign-In Service, EMail Service, SMS Service, and Message Service APIs*** are designed and implemented as part of this implementation can be reused by any other application/service context.

7. **API Gateway Pattern** – The API Gateway pattern is used to:
   a. Avoid any point-to-point integration of the APIs with the Applications that imposes direct and complex dependencies between the service consumers and the providers and creates a very inflexible and unmaintainable architecture. The API Gateway creates a layer of abstraction between the service consumers and the service providers. This eliminated direct dependencies between the service consumers and the service providers that makes the architecture flexible and maintainable.

   b. Support different types of client devices with different types of contents

   c. Route the service calls to the appropriate service providers

   d. *Signle Entry Point for all Service invocation* – provides a central point for all Security and compliance enforcements, service performance and SLA amanagement and monitoring etc.

8. **Open Standards and Open Source Based –** The Application is implemented using vasious open standards and open source based tools as described in the following sections.

9. **NIEM 3.0 Compliance –** The data model used for SafeKids is NIEM 3.0 compliant. The Person Data Model is compliant to the NIEM Core Schema Model.

2.2 Architecture Layers

This section describes various architecture layers for the SafeKids N-Tier distributed architecture.

2.2.1 User Layer

This layer consists of varous users who access various business functions provided by the SafeKids application. For the SafeKids prototype, the following three key users are identified:

- Biological/Adoption Parents;
- Foster Parents; and
- Case Worker.

Please note that, in the current prototype scope, the Bilogical/Adoption Paretns can only send mssages through private mailbox to the Foster Parents and the Case Worker. The Foster Parents and the Case Worker can't send messages to the Biological Parent.

2.2.2 Acces Channels

The SafeKids users can access the application functions through following access channels:

- **Web Browser** – users can acces the application functions through PC/Notebook computers using web browser through internet. The responsive design of the user interface allows the users to access the SafeKids application through almost all standard web browsers.
    - Google Chrome;
    - Mozzila FireFox;
    - MS Internet Explorer/Edge;
    - Opera;
    - Safari on Mac;
    - Safari on Windows; and
    - Safari on Mac.

- **Mobile Devices** – users can access the SafeKids application function through any mobile devices (tablets/cell phones) on major standard mobile platforms:
    - Android;
    - iOS; and
    - Microsoft etc.

2.2.3 Application Layer

The Application in the current prototype scope provides the following functions to the Biological/Adoption Parents:

- Sign-In;
- User Registration;
- Parent Profile Management;
- Child(ren) profile Management;
- Facility Search; and
- Messaging to Case Worker and Foster Parents through Private Mailbox.

Pease refer to the following deliverable for detailed business requirements interms of User Stories.

- *ADPQ-SafeKids-UserStories*

The SafKids Application uses Model-View-Controller (MVC) architecture. The The application is implemented using tools/technologies that are completely technology platform (such as Java/JEE, MS .NET etc) agonstic. The application is implemented using the following key Open Source/Open Standards:

- AngularJS;
- JSON;
- Twitter Bootstrap;
- HTML5; and
- CSS3.

As described above, following are the key application features:

- Responsive Design – optimal user experience through a wide range of devices;
- Section 508 complaint;
- Technology Platform Agnstic; and
- Flexible and maintainable design using Model-View-Controller (MVC) Pattrern.

2.2.4 Services Layer

The Service layer of the SfeKids is based on N-Tier distributed architecture that delivers various business and technical services to the Application through a central *API Gateway*. As dscribed in Section 2-1, the business and technical services requirements are derived from the *User Stories*. The business and technical services are implemented using *Microservices* architecture as *RestFul APIs*. The APIs are deployed in IIS 7.0 web server.

The Application Layer invokes the services through the API Gateway and send HttpRequstMessage with the request data elements as JSON objects. The API Gateway, then route the service call to the appropriate business or technical service. After the service is executed, the API Gateway returns the HttpResponseMessage with the response data as JSON object to the Application.

The high level architecture for the Microservics is shown in the following Figure 2-2.

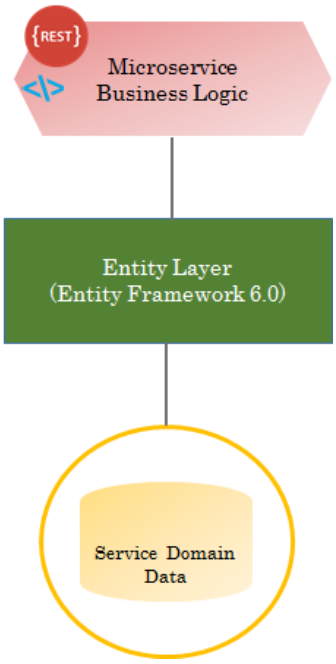

Figure 2-2 Microservice Logical Architecture

The SafeKids data is decomponsed into multiple data domains such as Security, Person, and Message etc. The Security data domain holds the user authentication and authorization infomaton, the Person data doman holds the Person (Child, Parent etc.) demographic data and the Message data domain holds the message data between the Biological Parent and the Case Worker and the Foster Parents. One Microservice is built for each data domain where, the Microservice has the complete authority to perform CRUD (Create, Read, Update, Delete) operations on the specific set of domain data. No other services can perform CRUD operation on this domain data set. This makes the Microservices *Autonomous API.* In the Microsevice Architecture (Figure 2-2), the business logic layer implements all the business ligic for the service. The Entity Layer integrates the service domain objects to the corresponding database entities for CRUD operations.

The Microservice Business Logic Layer interacts with the physical database through Object-Relational Mapping (OR Mapping) using .NET Entity Framework 6.0 and LINQ Query.

Key Tools and Technologis used in the Service Layer include:

- Microservice Architecture
- RESTful API
- ASP.NET MVC 4.0
- C#
- Entity Framework 6.0 with LINQ Query
- IIS 7.0
- Windows Server 2008 R2

The details of the Business and Technical sevices are provided in Section 4.

2.2.5 Data Layer

The Data Layer provides persistence storage for the Application data. The Data model is compliant to the NIEM-Core schema model.

The physical database is implemented on MS SQL Server 2008

As shown in the Figure 2-1, the Data Layer consists of the following three categories of data:

- **Person**– to store Parent and Children demographic data
- **Message** – to store communication messages between the Parent of the Foster Kid and Case Worker and the Foster Parent.
- **Security** – to store user authentication data

The Logical Data Model (LDM) for SafeKids (for the prototype scope) is shown in Figure 2-3. The LDM (Object Model) is developed using UML Class Diagram in Sparx Enterprise Architect.

Tools and Techmnologies:

- Windows Server 2008 R2;
- Microsoft SQL Server 2008; and
- Transact SQL (T-SQL).

The Microservice Business Logic Layer interacts with the physical database through Object-Relational Mapping (OR Mapping) using .NET Entity Framework 6.0 and LINQ Query.

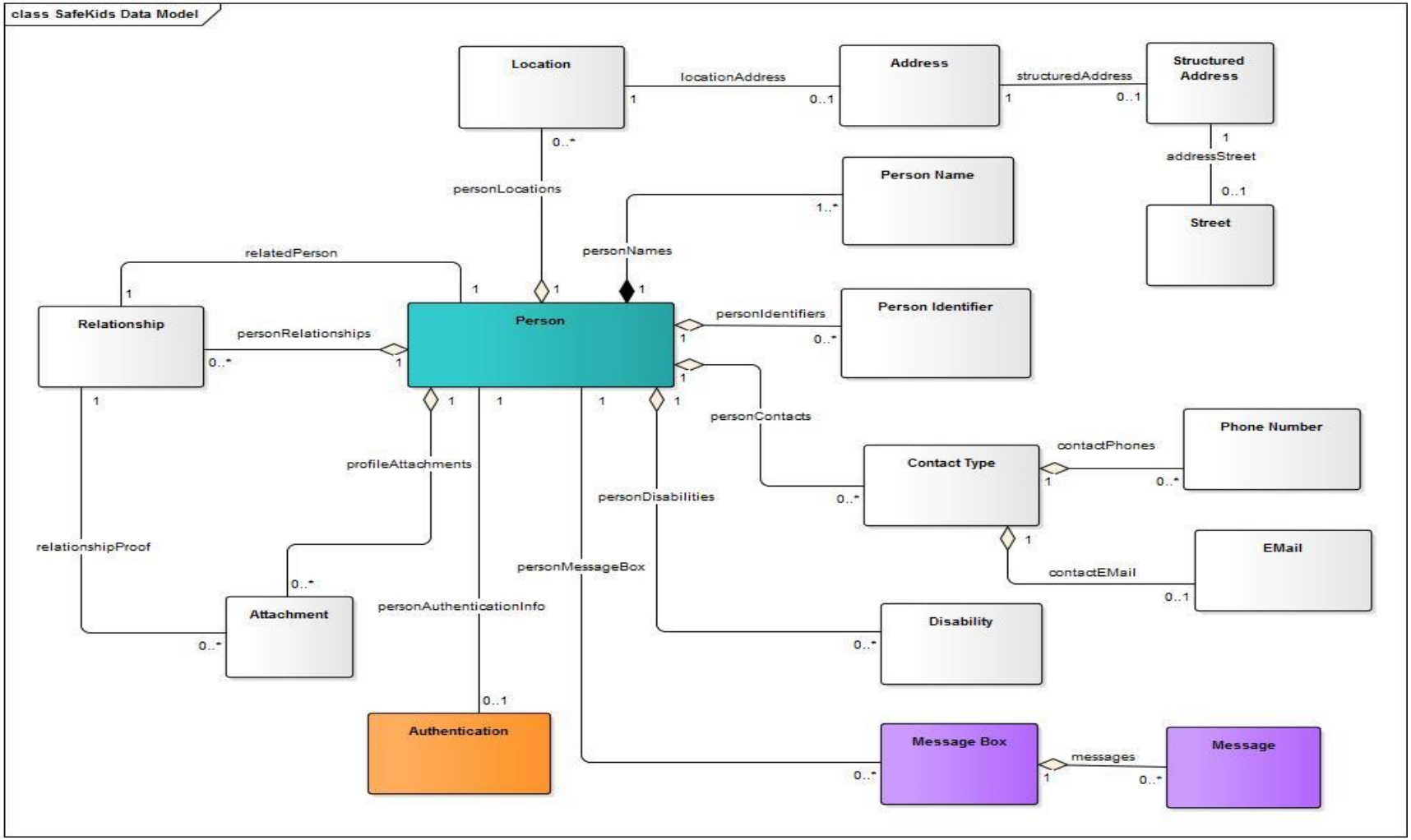

Figure 2-3 SafeKids Logical Data Model (LDM)

3 Physical Architecture

The application is deployed on an IaaS infrastructure using Amazon Web Services (AWS) as a provider. As shown in Figure 3-1, the application EC2 hosts the prototype application, SafeKids.

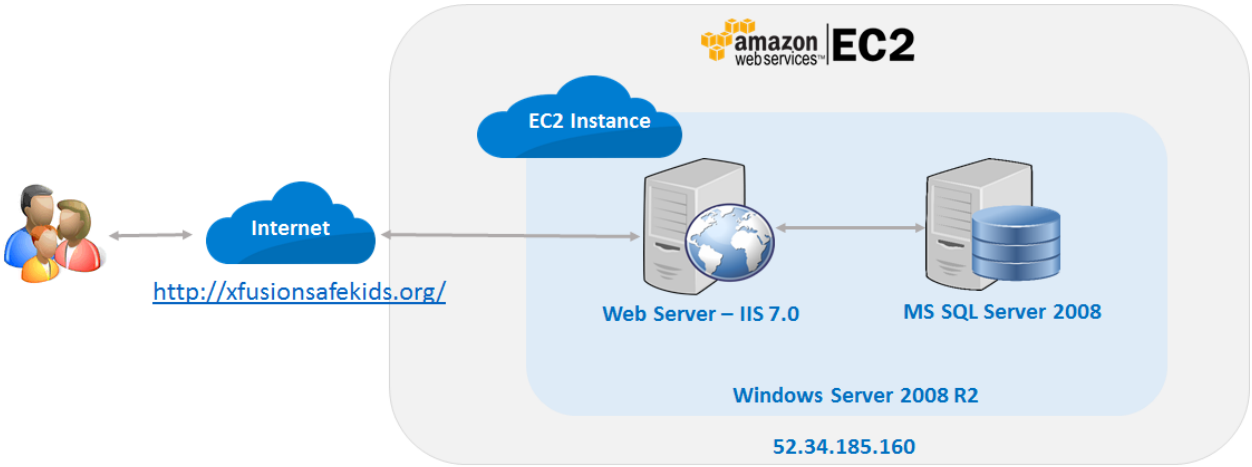

Figure 3-1 Physical Architecture – Production Environment for the Prototype

The Application and the database server for the prototype application are running on the same EC2 instance running MS Windows Server 2008 R2. The Application runs on the web server, IIS 7.0.

NOTE: The prototype application is deployed on a single server, and doesn't depict the horizontal scalability feature through load balancing. For the actual production environment, the application architecture will support horizontal scalability through load balancing.

3.1 Modern and Open Source Technologies

1. AngularJS - Front-end MVC framework to create a single page application
2. Twitter Bootstrap - A popular CSS framework
3. HTML 5
4. JSON
5. REST API
6. Google Map API
7. Selenium (for automated Regression and Load testing)
8. XML
9. GitHub – Source Code Control/Versioning
10. Model-View-Controller (MVC) Pattern
11. API Gateway Pattern
12. Object-Relational (OR) Mapping – .NET Entity Framework 6.0
13. NIEM CORE Schema Model

Additional technologies are listed in LICENSES.md on GitHub.

4 Microservices API Specification

This section provides high level descriptions of the Microservices API implemented for this application. The technical and the Business Services re dscribed below.

4.1 Technical Services

The Technical Services are the implementation of the technical functions/capabilities that are used by the business services to implement business functions and are business context agnostic. Following Technical Services are implemented.

EMail Service: RESTful API, used to end email to a recipient using SMTP. In the SafeKids application the EMail service is used to send the Verification Code for user registration via email.

SMS Service: RESTful API, used to end text message to a recipient using SMS. In the SafeKids application the SMS service is used to send the Verification Code for user registration via SMS Message.

Message Service: RESTful API, used to communicate with other parties through a privae mainbox. In the SafeKids application, this service is used by the Biological Parents to communicate to the Foster Parents and the Case Worker using a private mailbox. The message data is stored in the Message Data Domain tables in the database and the Message Service performs CRUD operation on this data set (Figure 4-1).

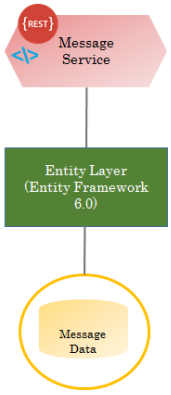

Figure 4-1 Message Service

Sign-In Service: RESTful API, used to for user authentication. In the SafeKids application, this service is used to authenticate the Biological Parents. The user authentication data is stored in the Security Data Domain tables in the database and the Sign-In Service performs CRUD operation on this data set (Figure 4-2).

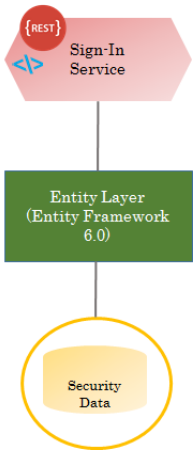

Figure 4-2 Sign-In Service

4.2 Business Services

Business Services are the implementations of specific business functions/capabilities. The Business Services use Technical Services for the service implementation. The Business Services are used using Microservices Architecture as RESTful API.

Person Profile Service: RESTful API, used to create and update Person Profile such as Biological Parent, Foster Parent, Case Worker, and Foster Child etc. A high level view of the Person Profile Service model is shown in Figure 4-3.

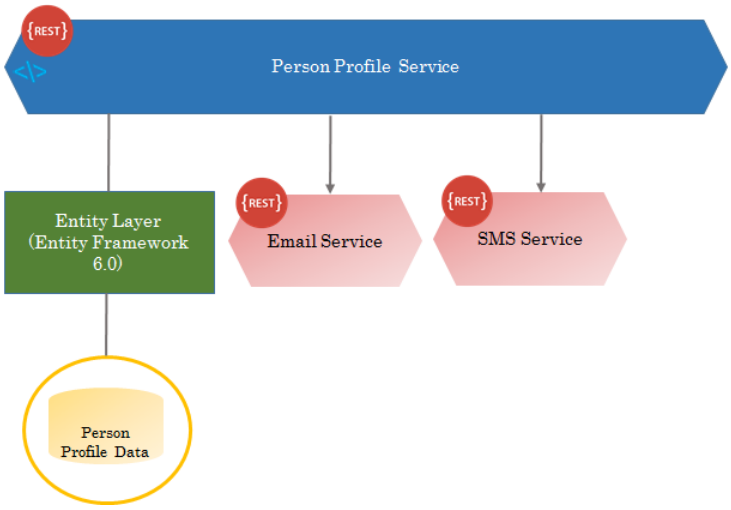

Figure 4-3 Person Profile Service

In the SafeKids application, this service implements the following functions:

- User Registration – registration of new Biological Parents. This API uses the folloing two Technical Services for registration code verification.
  - *EMail Service* – for sending registration verification code via email
  - *SMS Service* – for sending registration verification code via SMS Text Message
- Maintain Parent Profile – Create and update Parent Profile
- Maintain Child(ren) Profile – Create and update Child(ren) Profile

Facility Search Service: RESTful API, used to perform nearby Foster Care Facility search by ZIP code (Figure 4-4).

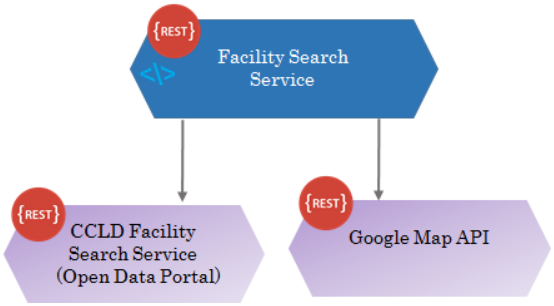

Figure 4-4 Facility Search Service

This API uses the following two external APIs.

- *Community Care Licensing Division (CCLD) Facility Search Service* - this service is provided by CDSS. This service returns nearby Foster Care Facilities and associated details for a specific Zip Code.
- *Google Map API* – used to display visually, the Foster Care facilities on a Map.

5 Continuous Integration and Deployment

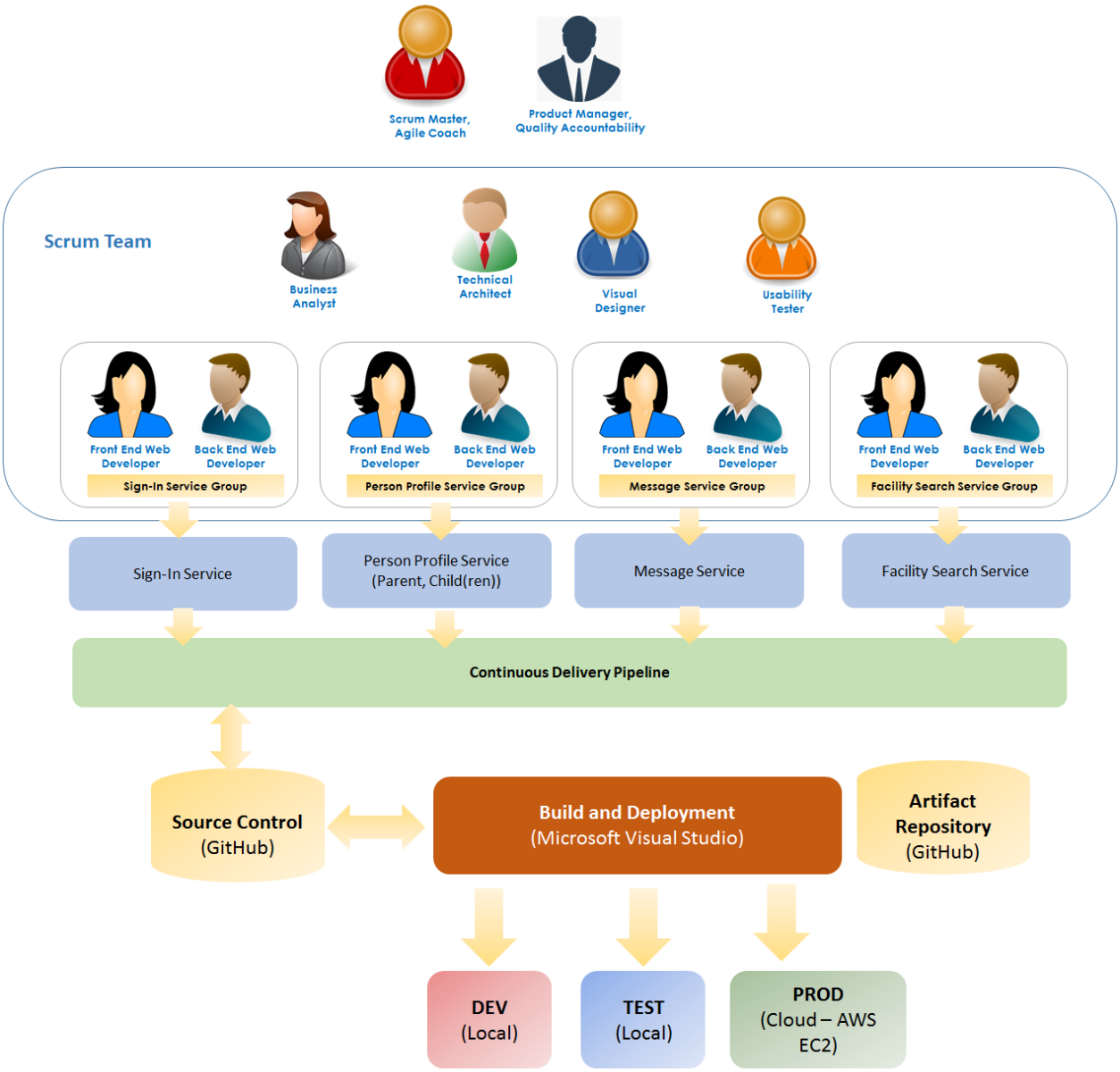

Figure 5-1 Continuous Delivery Pipeline

As shown in Figure 5-1, the Scrum Team is organized to work on four application features in parallel in an iterative and incremental fashion. The work product from each team is checked-in to the source code control system (GitHub) twice a day – once at the middle of the day and once at the end of the day.

The built is performed after each check-in using Microsoft Visual Studio and automated regression test is performed on the new built. If the regression test is successful, then the built is deployed in the DEV/TEST server where integration testing is performed. If the regression test is failed, a notification is sent to the developers. The responsible developer fix the code, performs unit testing of the code and checks-in the code to GitHub. At this time, a buid is performed and regression test is run again. Integration Testing is performed in the TEST environment. After successful testing in the TEST environment, the built is deployed in the PROD environment in the AWS environment.

6 Configuration Management

GitHub is used for source code control and as document repository and version control system.