**High Level Design**

Image result for instant edge logo

**The Enterprise Transformation Platform**

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

|  |  |
| --- | --- |
| Document Reference: | ISS\_IE\_HLD\_1.0 |
| Client: | **I**nstant **E**dge |
| Project: | **I**nstant **E**dge- Manage Operations Module |
| Document Title: | High Level Design |
| Version | 1.0 |
| Date | 20/10/2016 |
| Author | Zhao Pengcheng |

**Amendment History:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Review & Approval Form Number | Date | Brief description of amendments and affected pages, paragraph | Author |
| 1.0 | ISS/IE/TS/v1.0 | 20/10/2016 | Initial version | Zhao Pengcheng |

**Team Members:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Group** | **Matriculation No.** |
| Rameswari Mohanty | SE24/4FT | A0148532E |
| Vrinda Gupta | SE24/4FT | A0148518X |
| Vignesh Selvaraju | SE24/4FT | A0148576N |
| Zhao Pengcheng | SE24/4FT | A0148528W |

Table of Contents

[1. Introduction 4](#_Toc468866936)

[1.1 Purpose 5](#_Toc468866937)

[2. High Level Architecture 5](#_Toc468866938)

[2.1 Overview 5](#_Toc468866939)

[2.2 Programming Language 7](#_Toc468866940)

[**2.2.1** **Syntax** 7](#_Toc468866941)

[**2.2.2** **Packaging** 7](#_Toc468866942)

[**2.2.3** **Object Lifecycle** 7](#_Toc468866943)

[2.3 Transition Strategy 7](#_Toc468866944)

[**2.3.1** **Entity Objects** 7](#_Toc468866945)

[**2.3.2** **Controller Objects** 8](#_Toc468866946)

[**2.3.3** **Boundary Objects** 8](#_Toc468866947)

[**2.3.4** **Illustration of Transition Strategy** 8](#_Toc468866948)

[2.4 Backend Architecture 12](#_Toc468866949)

[2.5 Presentation Layer 12](#_Toc468866951)

[**2.5.1** **Screen Layout** 12](#_Toc468866952)

[**2.5.2** **Style Classes** 13](#_Toc468866953)

[**2.5.3** **Impacted Analysis objects** 13](#_Toc468866954)

[**2.5.4** **Impacted Use Cases** 13](#_Toc468866955)

[2.6 Data Access Layer 13](#_Toc468866956)

[2.6.1 JPA Architecture 14](#_Toc468866957)

[2.6.2 Important Elements in Data Access Layer 14](#_Toc468866958)

[2.6.3 Impacted Analysis objects 14](#_Toc468866959)

[2.6.4 Object Relational Mapping 15](#_Toc468866960)

[2.6.5 Impacted Use Cases 17](#_Toc468866961)

[2.7 Business Layer 18](#_Toc468866962)

[**2.7.1** **Impacted Analysis Objects** 18](#_Toc468866963)

[**2.7.2** **Impacted Use Cases** 18](#_Toc468866964)

[2.8 Usability 18](#_Toc468866965)

[2.9 Performance, Reliability and Capacity 18](#_Toc468866966)

[2.10 Security 18](#_Toc468866967)

[3. Deployment 18](#_Toc468866968)

1. **Introduction**

Enterprise IT organizations, their systems integrators and software vendors have been reasonably good in the past more than 15 years to help business organizations integrate their disparate commercial processes into a (more) holistic whole – while at the same time neglecting own (management) decision support processes and systems.

Instant Edge provides a transformation platform that addresses this problem space by helping executives, line- and project managers, team leads, members, and employees – essentially any stakeholder of a business transformation initiative to drastically improve motivation of actors within the corporation, enable better decision making, avoid failed transformations and dramatically improve quality of delivery – achieving the desired real business partnering status with on par business interactions.

The Instant Edge Platform is structured with following business process areas:

1. Ensure Benefits delivery
2. Manage Changes
3. Manage Programs and Projects
4. **Manage Operations**
5. Manage Organizations

Instant Edge requires the development of Manage Operations Process Area. This involves the development of following Key components:

1. Manage Service Portfolio
2. Manage Service Catalogue
3. Manage Service Desk
4. Manage Service Level
5. Manage Incident
6. Service Validation and Testing
7. Manage Application
8. Manage Knowledge
9. Manage Mobile Integration
   1. **Purpose**

This document serves as a guideline for carrying out design modelling of Manage Operations Module of Instant Edge Platform. It describes the Software Architecture for Manage Operations Module of Instant Edge Platform including the Non-functional requirements to be considered.

1. **High Level Architecture**
   1. **Overview**

The target implementation platform is chosen to be **Java EE 7**. The following diagram illustrates the architecture layers.

****

The following architectural decisions are made:

* The system is a web-based application with 3 layers: Presentation, Business and Data Access
* The Presentation Layer will be implemented using Java Server Faces with Backing Beans as Controllers, Prime faces as JSF Component Library.
* The Business Layer will be implemented using Enterprise JavaBeans.
* The Data Access Layer will be implemented using Java Persistence API.
* Data are persisted in MySQL.

The following subsections capture the changes required to transition from analysis to design in terms of:

* Programming Language
* Transition Strategy
* Presentation Layer
* Data Access Layer
* Business Layer

Further subsections discuss strategies in dealing with the following nonfunctional requirements:

* Usability
* Performance, Reliability and Capacity
* Security
  1. **Programming Language**

**Java** is the programming language of choice for Instant Edge Platform.

* + 1. **Syntax**

Unless otherwise specified, design and implementation should be performed following the coding convention captured in Java® Coding Style Guide found at http://www.oracle.com/technetwork/java/codeconvtoc-136057.html.

* + 1. **Packaging**

The classes and interfaces for Manage Operations Module of Instant Edge Platform are packaged as follows.

|  |  |
| --- | --- |
| **Package** | **Contains the following classes/interfaces** |
| entity.ops | All JPA Entity classes |
| jpa.service.local | All EJB interfaces |
| jpa.service.exception | All EJB exceptions |
| jpa.ops | All EJB Implementations |
| jsf.ops | All Backing beans |
| jsf.converter | All Converters |

* + 1. **Object Lifecycle**

A Java object is instantiated by calling its constructor with the new keyword. There is no need to deal with the destruction of object in Java. An instantiation interaction can be shown in sequence diagram using constructor name as message name.

If the instantiating object requires a reference to the instantiated object, it is generally realized by passing in the object reference of the instantiated object in the constructor of the instantiating object.

Other objects like backing beans and enterprise java beans, their lifecycle will be managed by the application server.

* 1. **Transition Strategy**
     1. **Entity Objects**

All the entity objects in the analysis diagram will be represented as JPA Entity objects in the design diagram. The list entity objects in the analysis diagram will be represented as DAO object in design diagram. Add and remove, retrieve and find operations on the list entity objects in the design diagram will be performed on the corresponding DAO object. After modifying the attributes of the entity objects in order to persist the entity update method will be called in the corresponding DAO object.

* + 1. **Controller Objects**

All the controller objects in the analysis diagram will be JSF Managed Beans and EJBs. EJB perform all the data access functions and JSF Managed Beans delegate the functionality to EJB methods. The Backing Bean calls this EJB method when a user action is performed. The Controller methods which performs validations which does not involve entities can be performed JSF Backing Bean methods if applicable.

* + 1. **Boundary Objects**

All the boundary objects in the analysis diagram will be transformed to xhtml pages. The messages passed between the actor and the boundary objects will be represented as the messages between actor and the xhtml pages. Form validations performed by the boundary object will be performed by the xhtml and java script code generated by the JSF framework whose operations will not be shown in the design diagram.

* + 1. **Illustration of Transition Strategy**
       1. **Static Structure in Analysis Diagram**



* + - 1. **Dynamic interaction in Analysis Diagram**



* + - 1. **Static Structure in Design Diagram**

****

* + - 1. **Dynamic Structure in Design Diagram**

****

* + - 1. **Invocation of Abstract User Case**
         1. **Analysis Model**



* + - * 1. **Design Model**

In **Java EE 7**, the invocation of abstract use case is implemented using CDI, the dependency between controllers (Backing beans) is handled by the application server. The class diagram is shown below.



* 1. **Backend Architecture**

The Backend Architecture after applying transition strategy is shown below:

****

* 1. **Presentation Layer**

The User Interface of Instant Edge Platform is implemented in Java EE using JSF and Prime faces. The web pages will communicate with the business functions through the Backing Beans. To improve the look and feel of the User Interface we will specify styles to the components within the web pages using Cascading Style Sheets (CSS). Prime faces theme will be used to enhance the look and feel of the application. We will make use of JSF templates and Composite Components to prepare the screen layout and to create re-usable components which can be used throughout the application.

* + 1. **Screen Layout**

The screen in the Instant Edge Platform has been divided in three sections namely:

**TopBar**

The TopBar section displays the company logo and the banner with a welcome message for the logged-in user.

**LayoutMenu**

The Left Menu section displays the navigation menu for the user to navigate to various pages. The options available to the user depend on the role and priveledges of the user.

**Content**

The Content section displays the relevant page as per the context.

The layout of the screen is described in the figure below.



* + 1. **Style Classes**

Prime Faces components are used in the application. Default styles of Prime Faces can be replaced by Cascading Style Sheets (CSS) if have been used to style individual components on the user interface.

* + 1. **Impacted Analysis objects**

All the boundary objects are impacted. They translate into XHTML pages to display information and accept user actions supported by the Backing Beans. The data required for presentation will be available in the Backing Bean. Backing bean can be either request scoped, view scoped, session scoped or application scoped as per the requirement.

The Backing Beans are responsible for performing operations corresponding to the displaying of pages, reception of user input, calling the corresponding business operations, etc. Validation of forms on the client side will be performed by the java scripts created by the JSF framework and validation of forms at the server side are performed at the Backing Beans.

* + 1. **Impacted Use Cases**

All the use cases that perform user interaction are impacted.

* 1. **Data Access Layer**

### JPA Architecture

The Java Persistence API is used for persisting the objects in this Project. The JPA is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational database. JPA allows an object's object-relational mappings to be defined through standard annotations or XML defining how the Java class maps to a relational database table. JPA also defines a runtime Entity Manager API for processing queries and transaction on the objects against the database. JPA defines an object-level query language, JPQL, to allow querying of the objects from the database.

### Important Elements in Data Access Layer

**Entities**

A persistence entity is a lightweight Java class whose state is typically persisted to a table in a relational database. Instances of such an entity correspond to individual rows in the table. Entities typically have relationships with other entities, and these relationships are expressed through object/relational metadata. Object/relational metadata can be specified directly in the entity class file by using annotations, or in a separate XML descriptor file distributed with the application.

**Entity Manager**

This is an interface used to interact with the persistence context. An Entity Manager instance is associated with a persistence context. Within the persistence context, the entity instances and their lifecycle are managed. The Entity Manager methods are used to create and remove persistent entity instances, to find entities by their primary key, and to query over entities. The set of entities that can be managed by a given EntityManager instance is defined by a persistence unit. A persistence unit defines the set of all classes that are related or grouped by the application, and which must be associated in their mapping to a single database.

**EntityManagerFactory**

The EntityManagerFactory interface is used by the application to obtain an application-managed entity manager. When the application has finished using the entity manager factory, or at application shutdown, the application should close the entity manager factory. Once an EntityManagerFactory has been closed, all its entity managers are considered to be in the closed state.

**Persistence.xml**

The persistence.xml file gives you complete flexibility to configure the EntityManager. The persistence.xml file is a standard configuration file in JPA. It has to be included in the META-INF directory inside the JAR file that contains the entity beans.

### Impacted Analysis objects

All the controllers and entity objects are impacted. The Entity objects will be translated as JPA Entity Objects. The relationship between the entity objects are retained in the JPA Entity Objects with proper annotations. The owned object/ object collection and the associated object/object collection will be defined as similar to the analysis diagram in the JPA Entities. These collection objects will be loaded by the lazy loading mechanism managed by JPA.

### Object Relational Mapping

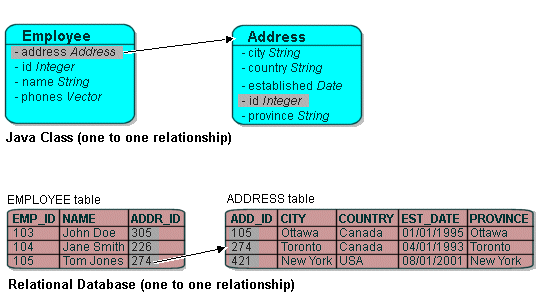
**Non Entity Objects Collection in a Single Class Mapping**

Collection of objects such as String, Integer, Double, etc. in a Single Object will be mapped to two tables. One main table representing the single object and another table representing the primitive object collection with the key of the main table as the foreign key in the collection table.

**One to One Association Relationship Mapping**

One to one assocation relationship between two entities will be mapped to two tables representing each entities with the primary key of one table placed as foreign key in the other table based on the direction of association.

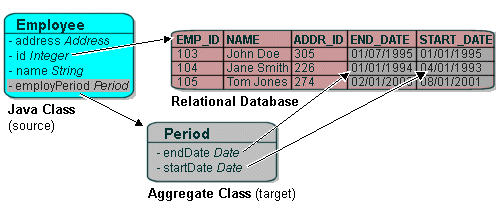
**Example**



**One to One Aggregation Relationship Mapping**

One to one aggregation relationship between two entities will be mapped to single table in the database.

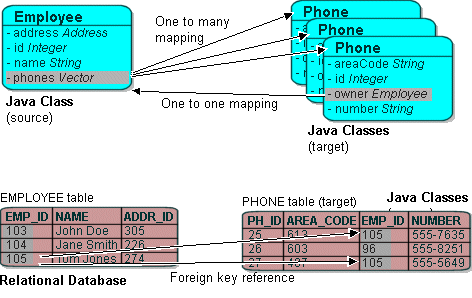
**Example**



**One to Many Aggregation/Association Relationship Mapping**

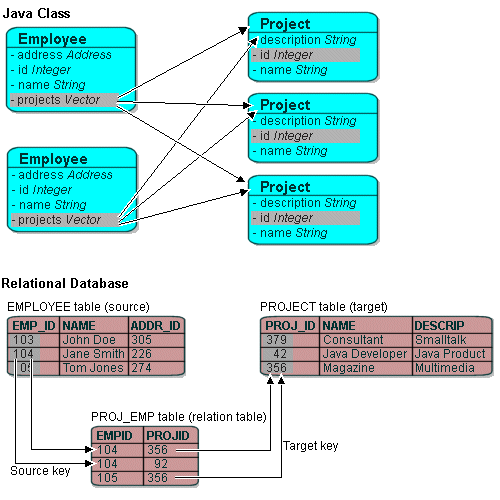
One to many aggregation relationship between entities will be mapped to two tables with representing each entities with the primary key of one table placed as foreign key in the other table based on the direction of association.

**Example**



**Many to Many Aggregation/Association Relationship Mapping**

Many to many aggregation relationship between entities will be mapped to three tables with two tables representing each entities and the relationship between them is placed in the other table with the primary key of both the entities.



### Impacted Use Cases

All the use cases that perform database related operations are impacted.

* 1. **Business Layer**

The Business Layer for Instant Edge Platform performs the key business functionalities. This is implemented as an intermediate layer between the Presentation and Data Access Layers. Any Business logic (excluding Presentation logic) that requires manipulating data or otherwise (flow control and business logic) are encapsulated in this layer. Typically, classes in this layer will make use of one or more entity classes. The JSF Backing Bean classes along with the EJB Classes perform the business logic in our application and the database operations are performed by using the appropriate entity Objects.

* + 1. **Impacted Analysis Objects**

All the control objects that encapsulate business logic and/or interact with persistent entity objects are impacted. The methods which interacts with the DAO will be placed in the EJB Objects whereas the methods which perform business specific validation could be placed in the Backing Bean.

* + 1. **Impacted Use Cases**

All the use cases that perform business logic and/or involve data persistency are impacted.

* 1. **Usability**

The usability of Instant Edge Platform is fulfilled using web-based user interface as designed in the above subsection on Presentation Layer with latest JSF 2.1 and Ajax prime faces.

* 1. **Performance, Reliability and Capacity**

By using Java EE 7 platform, the application servers provide load balancing and clustering thereby the performance, reliability and capacity of the system is managed by the server that doesn’t affect the software design. This is achieved through the system configuration of hardware and platform.

* 1. **Security**

The security of Acumen PP is realized using the Apache Shiro framework which realizes the JAAS.

1. **Deployment**

The Deployment of the System will be done according to the following diagrammatic representation.



1. The Application Server will be containing the WAR of the Application which includes the Web and the EJB modules. The Server we are using is glassfish.
2. The database server contains the MySQL server independently which is capable of clustering.