**Software design specifications**

**Version 1.0**

**Ronak Jain**

**Saurav Singh**

**Shiv Bajpai**

**Siddhant Misra**

**Raaghav Mehta**

1. **Introduction**

**1.1 Purpose of this document**

The purpose of this software design document is to provide a low-level description of the agile based product development system, providing insight into the structure and design of each component.

In short, this document will provide you with all the insights of our product with a proper understanding of the inner working of Agile-PDS.

**1.2 Scope of the development project**

Agile product development system in present scenario is gaining its popularity and the good thing about this is it doesn’t only consider software development, this can be used as a global software which will help in ease of communication and increase iterativeness.

This change will help more and more industries of any field to use this product and will facilitate better understanding of development to their customers and even their employees.

Suppose one software company got any contract from the customer for building up of any particular software and then this will work efficiently and help in manage the development of particular software by dividing the work in modules and giving that particular work to all the different departments present in the company’s organizational structure.

Some of the major changes in this technology are:

1. Collaboration and interaction over rigid rules and lines of authority.

2. Involvement of customers and other stakeholders in the development process.

3. Prototypes are given more importance than the documentation.

4. Ability to respond to change.

**1.3 Definitions, Acronyms And Abbreviation**

* Data Objects – Data objects are Java objects with predefined structures capable of holding data in a structure that is quickly and easily accessible by other parts of the software system. They provide also can help provide a convenient abstraction of the data in a database so that it can be retrieved into a format, such as a denormalized format, that makes access and manipulation of the data easier than if the database had to be called directly.
* Denormalized - Normalization of a database is the activity of restructuring the database to avoid data anomalies and inconsistencies by focusing on functional dependencies to help structure the data. Denormalization is the act of undoing some of the structural changes made during normalization to help with performance.
* Digital Signature – A digital signature is a unique object which is strongly tied to a single entity and the document which signature is intended for. In the same way that an ink on paper signature has characteristics that are unique to a person due to variations in writing a digital signature has characteristics that uniquely tie it to a single person and signing instance.
* Document Interaction Class, XML Document Interaction Engine – These are the two terms that will be used to refer to the main software class described within this document.
* Editable Form Layout- A user interface presentation layout in which the contents of a document are presented to a user in the format of a form predefined editable areas based on the type of document which is being edited. This type of layout allows for changes to be made in a specific manner so that the data used in the form can be reassembled into a structured data format for transfer to other systems and archival.
* FOP Libraries – FOP stands for Formatting Objects Processor. The FOP Processor use an XSL-FO stylesheet and an XML instance to create PDF's, RTF's, and HTML files. FOP libraries bring the functionality of an FOP processor to a library form which can be used within another software program.
* JDBC/ODBC – These two acronyms stand for Java Database Connectivity and Open Database Connectivity API's which allow for standardized database access and interaction from software products.
* LegalXML – A standards body dedicated to issues related to the use of XML in the legal domain.
* PDF – Portable Document Format
* Pro se – This is a Latin term which directly translated means “for self” and is used to indicate that a party to a case has chosen to represent themselves to the court instead of choosing for an attorney to represent them to the court.
* Required Field – A critical field is a field in a data set for a document that is required for successful document generation. For example, missing parties in a case, missing county location of court, or other data elements that are required to create a valid legal document.

|  |  |
| --- | --- |
| XML Legal Document Utility | Version: <1.0> |
| Software Design Document | Date: 28-10-2016 |
| SDD-XLDU |  |

* Structured Data Format – A structured data format is data assembled into a discernible structure, such as when data is placed into an XML instance which is validated through the use of an XML schema which defines the structure of the XML document.
* UUID – Universally Unique Identifier. A UUID is an identifier standard in software construction which allows for generating identifiers which do not overlap or conflict with other identifiers which were previously created even without knowledge of the other identifiers.
* Workflow – The movement of documents through a work process that is structured into tasks with designated persons or systems to perform them and the definition of the order or pathway from start to finish for the work process.
* XML – eXtensible Markup Language
* XSL – XML Stylesheet Language, which is used to transform and specify formatting for presentations of XML instances. XSL is a family of specifications that include XSLT, XSLFO, and XPath. XSLT stands for XSL Transform, which is used to transform an XML instance from one form to another. XSL-FO stands for XSL Formatting Objects, which is a specification for formatting objects which format the output of presentations of XML instances in forms such as RTF type files, PDF type files, or HTML files. XPath stands for XML Path Language and is a specification for accessing parts of an XML document using the path to the part in the hierarchy of the XML instance.

**1.4 References**

We referred to our SRS document for the description of the modules and for making sequence and class diagrams.

**1.5 Overview of the document**

The Software Design Document is divided into 11 sections with various subsections. The sections of the Software Design Document are:

1. Introduction
2. Glossary
3. Use Cases
4. Design Overview
5. System Object Model
6. Object Descriptions
7. Object Collaborations
8. Data Design
9. Dynamic Model
10. Non-functional Requirements
11. Supplementary Documentation

**2. System Architecture Description**

**2.1 Overview of Modules/Components**

**UI 1:** The XYZ screen will display a login page with a drop box, where there can be three different types of personnel’s entering their respective username and password.

**UI (Customers):** The customers can see the progress and the timeline for the respective project. Also there will be a chat system, if there are any messages for any of the teams, the same can be conveyed through the chat system.

**UI (Manager):** The manager/HRD can create a new project, set the modules, assign teams to these modules and specify the timeline within which all this work is supposed to be done.

**UI (Development team):** The personnel in the Development team can update their project progress/report, time to time. They can also send and revert messages through the chat system, if needed.

**2.2 Structure and Relationships**

**ASM1:** The final product made by the firm is feasible according to their standards.

**DEN1:** The operation of Agile-PDS depends on the firm’s Manager/HRD who is Responsible for the creation of the task and defining its modules, later

Assigning it to the product development team.

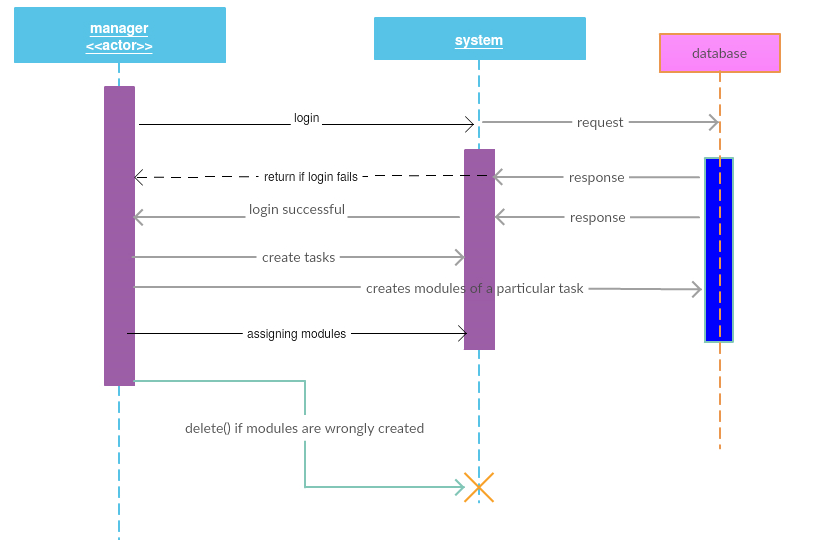
**DEN2:** The operation of Agile-PDS depends on the employees working in product Development team inputting their work progress which helps in framing the Timeline of the final product.

**DEN3:** The operation of Agile-PDS depends on the customer in regular Commentation and communication about the final product’s modules/goals.

**SEQUENCE DIAGRAM**

**1. Manager (actor)**

Manager is the actor who is responsible for the commencement of the product development phase, which he does using the system’s interface provided to him.

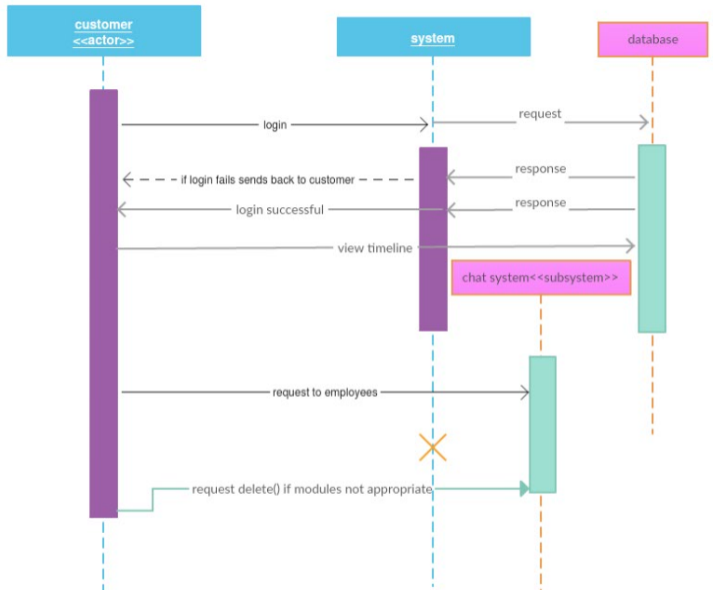


**2. Customer (Actor)**

A customer is the actor who is defining the product specification to the

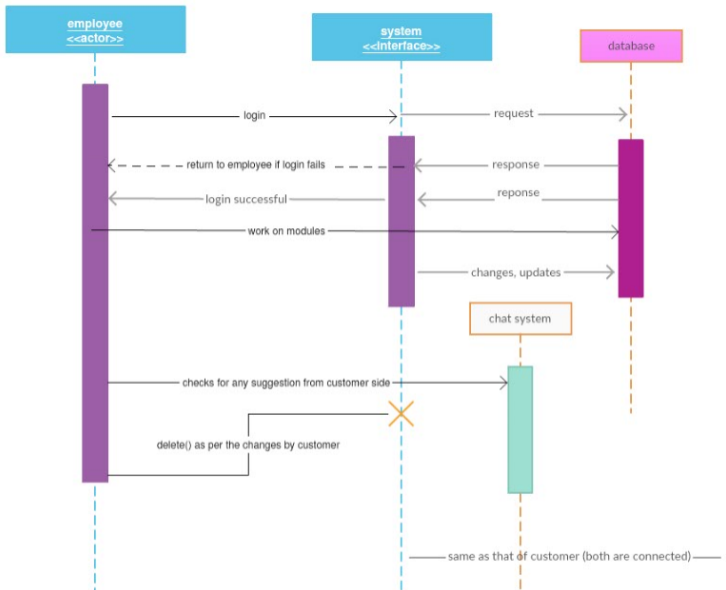
Manager, which in turn is defining modules for the product.

Customer can view the progress of his product and can suggest some changes which he requires to be done in the product.

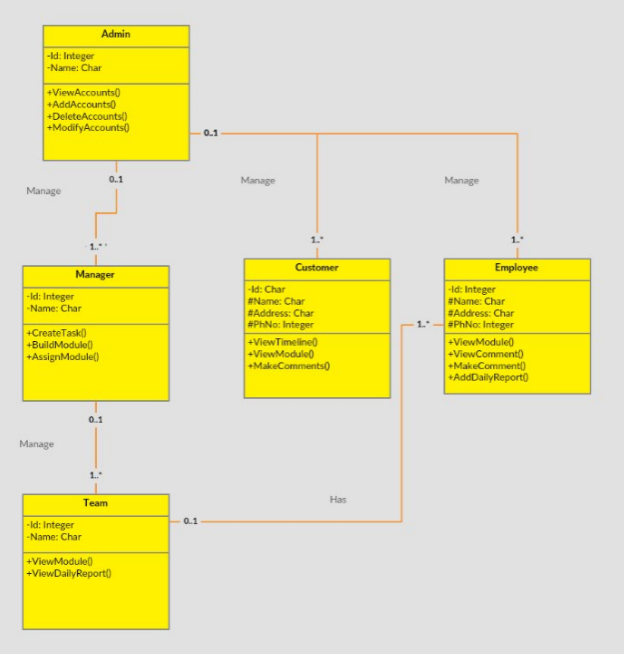


**3. Employee (actor)**

The employee working in the PDT is the actor who is responsible for the daily reporting on his assigned module by the Manager. He will be looking towards the changes required by the customer in the product and will help in maintaining proper product’s timeline on the weekly basis including updates, deletion or any modification done by them.



**Class diagram**



**2.3 User Interface Issues**

Well for the time-being there is no issue as our system is under development and will update as soon as it’s approved by the company.

**3. Design overview/ detailed description of components**

|  |
| --- |
| **task.name**: The name of the task must be decided and shall be posted by manager whenever he wish to create his new task and shall be shared among the PDT.  **Task.customer\_name:** the name of the customer who has given the task must be written while creating the modules.  **Task.company\_name :**name of the company to which customer belongs shall be provided. |
| **Task.create\_modules :** manager shall create the modules based on the product specification which that company requires.  **Task.assignment \_of\_product\_to\_PDT :** Manager shall assign the task as per the specification of the team. |
| **Task.timeline :** Basically this will let customer to see the whole levels that till what stage the product is finished and which team is currently working at that moment.  **Task.changes\_in\_modules :**  Customer shall ask for some changes if its not according to his requirements.  **Task.chat\_system :** Using this chat system customer can give update about the changes whether it's okay or not and relating to his response the team can look after that query. |
| **Task. Customized \_module :** employee shall work on the allotted module related to his profile.  **Task.weekly\_update:** employees should weekly update the modules on the basis of the **Task.chat\_system** according to the customer requirements. |

**4. Reuse and Relationships to other Products**

**4.1 How reuse is playing role in your product design**

Well if we look towards the overall issue of reuse then we will find it very important and a fundamental thing to implement in any design but in our design there is no reusability of any fundamental freeware that could be incorporated as we are developing this as a whole new Agile-PDS.

**5. Design Decisions and Trade-offs**

**6. Pseudocode for Components**