**TY. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 1

**ERP-farmers**

**Project Statement of Work**

***05/01/2019***

***Version 1.0***

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front-end** |
| 62 | **161259** | **Aniket Thaware** | **Back-End** |
| 73 | **161854** | **Rushabh Pahade** | **Leader** |

**Approved By: Mahesh R. Dube**

**33**

**Academic Year: 2018-19 Semester: II**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Title | **3** |
| 2 | Background | **3** |
| 3 | Objective | **3** |
| 4 | Definitions and Applicable Documents | **4** |
| 5 | Business and/or Technical Environment | **4** |
| 6 | Description and Scope Of Work | **5** |
| 7 | Deliverables | **5** |
| 9 | Approach and Methodology | **6** |

# TITLE

ERP-farmers is a web-application which provides platform to farmers to sell their production and increase their income and also provides information regarding their crops and farming practices and other agriculture products.

* 1. This System provide the daily Weather forecast so that farmers can take their decisions for their crops according to weather conditions.
  2. This System also provide Market to the farmers so that farmers can add their crop and quantity of crop and nearby Market shops can buy from that Market Portal.
  3. This system aimed at decreasing the mediator between farmers and market shop and provide direct communication between this two so that it will help in increasing the farmers income.

# BACKGROUND

Agriculture is the primary source of livelihood for about 58 percent of India’s population. Studies identified several causes for farmers suicide such as poor agricultural income, absence of alternative income opportunities, and lack of access to information related to agriculture as the chief cause for the desperate condition of farmers in the India. The ERP-farmers will prove to be an effective web-application to give the access to information related agriculture and alternative income opportunities through Market Portal which will help farmers to increase their income.

* 1. The mandate of the system is to help and satisfy the farmers by providing them with the best possible solution while also keeping a good financial structure as to make profits and maintains sustainability.
  2. Farmers, Market agents, and Normal people can make use of this product. The service will be open to all while there may be a few premium features for Market agents which can be accessed only upon paying for those services. All services are free for the farmer.
  3. The product will use web engineering process to project the output.
  4. This project will acquire a service to provide the solution to some unsolved problems of our farmers, which come under the domain Agriculture and Marketing.
  5. If needed, provide additional background/contextual information on the requirement and/or the organisation (Department/Faculty, Office, etc….).
  6. Identify the reason the organisation needs to source the requirement externally (i.e. via Contractors). The first determination required is whether the requirement can be satisfied in-house. This subsection should articulate why the organisation (Department/Faculty, Office, etc….) feels that a contract is needed to meet this requirement.

# OBJECTIVE

The objective describes the long-term goals of the systems. This system has the following objectives:

* 1. The system should be capable of providing good user interface for the Weather so that farmers will easily understand it and take their decisions.
  2. Objective is to provides best price for farmers crop.
  3. Objective is to provide more customers to the farmers.

# DEFINITIONS AND APPLICABLE DOCUMENTS

**Stock**: Catalogue can be configured to display the availability of each item as “In Stock” or “Out of Stock” of the product.

**Market Agent (Customer):** Customer are the people who will buy crop production from farmer.

**Farmers:** Farmers are the people who will add crop in web-app for selling to the market. And also use other extra features of the app i.e. Weather, Crop-Cycle.

**Client:** Client is the customer who will buying this software.

**Crop-Cycle:** Crop-Cycle consists of information related seeding, growth and harvesting process of each crop.

**Documents:**

1. <http://grails.asia/grails-tutorial-for-beginners/>
2. <https://www.mongodb.com/cloud/atlas>
3. <https://farmer.gov.in/>

# BUSINESS AND/OR TECHNICAL ENVIRONMENT

The web-app requires the following Business and Technical Environment to successfully commence in the stipulated time and resources.

* 1. The hours of operation will be independent as that of the organisation with weekly feedback given during the reporting time.
     1. The team will work Monday to Friday, 2hrs per day.
* This time will be utilised to work on completing the project documentation which will take up a major role in the initial weeks of the project.
* Later weeks will have more time invested in project planning and implementation with the documents having a lighter format.
* Time will be evenly utilised for Documentation, Planning, Execution, Testing &Debugging.
  + 1. Further work can be completed on weekends depending on the team/member’s convenience.
  1. From technical point of view the website can be use on any gadgets like computer, tablets, mobile, etc. The web-app will have regular updates about new functionalities.
     1. Internet connection will be required for the smooth functioning of the system.
     2. No memory required for user side as this system is web application And all data stored in the cloud.

# DESCRIPTION AND SCOPE OF WORK

The work that is to be done under “ERP-farmers” involves multiple steps:

* 1. Connect Farmers to the market agents and Customers.
  2. Separate login and signup for market and Farmers.
  3. Farmers can add their crop production for sell and check weather condition for their crops.
  4. Customers can buy foods from this platform.
  5. Farmers can see crop cycle for their crops and can use in their fields at each step to get good results.
  6. While dealing with the system, a deal has to made with databases, updating new information and taking feedback of customer whether they are satisfied by this system and any suggestion.

# DELIVERABLES

The system is in the initial stage of development and some of the deliverables may vary as the system continues to develop into a product. Amongst the contract deliverables are the core concept of the project which will not change in any case. The system will stay true to its vision and the only changes may be seen are the ones in the User Interface.

These are some of the deliverables that team can outlie at this stage of development. Each stage has its own challenges and will be given apt importance by the contractor.

|  |  |
| --- | --- |
| No. | Details |
| 1 | Statement of Work |
| 2 | Feature Set |
| 3 | SRS Document |
| 4 | Feasibility Study and Project Plan using AGILE |
| 5 | Sprint level planning activity |
| 6 | Sprint Plan and Sprint Design |
| 7 | Software Configuration Management Plan (SCMP) |
| 8 | Sprint Execution |
| 9 | Sprint Review and Sign- offs |

# APPROACH AND METHODOLOGY

* 1. Preparing proper documentation and getting the views of the team and organisation by creating proper SOW, Feature Set Document and SRS Document.
  2. A Feasibility Study will be performed depending on the features discussed between the team and organisation and a Project Plan will be drawn up.
  3. The Project will follow the Agile model and all the necessary steps will be taken as per industry standards.
  4. A Sprint Execution will be carried out in phases to finish the project in the stipulated time, this will be done with the help of a Sprint Design and Plan.
  5. A Software Configuration Management Plan (SCMP) will be presented to ensure consistency of the product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life.
  6. At the end of each sprint, the team will have produced a coded, tested and usable piece of software.
  7. The System will be reviewed by the concerned organisation and all the issues will be presented to the team.
  8. Upon resolution of these issues a final and formal sign-off will be suggested.

**T.Y. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 2

**ERP-farmers**

**Project Feature Set Description**

***12/01/2019***

***Version 1.0***

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front End** |
| 62 | **161259** | **Aniket Thaware** | **Backend** |
| 73 | **161854** | **Rushabh pahade** | **Leader** |

**Academic Year: 2018-19 Semester: II**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Project Vision | **8** |
| 2 | Project Mission | **8** |
| 3 | Project Scope | **8** |
| 4 | Goals | **11** |
| 5 | Feature Set | **12** |
| 6 | Stakeholders | **13** |
| 7 | Acceptance Criteria | **13** |

# 1. PROJECT VISION

To provide platform for the farmers where they can get good price for their crops and also they can see weather condition to take decisions for their crops and also to provide crop cycle to help them at each step in their farming.

# 2. PROJECT MISSION

The Mission Statement summarises the aim of this project and what it is trying to achieve. This is our project Mission:

1. The ERP-farmers is an web application mainly for farmers as well as Market-agents (Customer’s) to increase the farmer’s income. This web application will be presented to both farmer’s and customer’s in the form of website and mobile application also.
2. The ERP-farmers web-app makes crop production marketing process convenient for farmers as farmers can add their crop production in the app and it will be visible to all nearby markets so that market agents can give orders to the farmers.
3. The purpose of our system is to design and implement ERP software for farmers that will help to increase their income through market and also help in farming process.

# 3. PROJECT SCOPE

**‘ERP-farmers’** will be made to provide platform to the farmers where they will get help in their farming process through crop-cycle and weather and also get good price for their crops through market so that they can increase their income. This system will connect farmer’s direct to the Market there will be no mediator. So that will help farmers to get good price for their crops.

These are our project goals as defined by the team:

1. Build Farmer Profile
2. Build Market-agents (Customer’s) Profile
3. Perform Add crop production for selling
4. Customer buy crop production
5. Display weather
6. Display Crop-Cycle

# 4. GOALS

|  |  |  |
| --- | --- | --- |
| Goal-ID | Priority | Factors Addressed |
| 1 | 1 | Build Farmers Profile |
| Target Audience | Farmers |
| Driver | To make farmers profile |
| Description | Populating the whole database with farmer’s details |
| Response | The goal is to extract farmer data from database and use the information received. |
| Open Issues | Discussion and Revision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goal 1 Description: | | | | |
| Specific Test | | | | |
| Is ‘What’ identifiable? | Is the ‘Why’ clear? | Can ‘Who’ be identified? | ‘Where’ will it be performed? | ‘Which’ resources are needed? |
| Collecting data of Farmers. | To populate the whole database for storing farmer’s details. | Farmer will use data to login to this system. | Software engineer’s machines | Simple data entry |

|  |  |  |
| --- | --- | --- |
| Goal 1 Description: | | |
| Measurable Test | | |
| Is the end result quantifiable? | ‘Figure’ of Measurement | Has the goal a clear end date/point? |
| Yes, as the Farmers who have registered can be counted. | Number of Farmer details stored in the database. | Farmers can use the functions of this system if the details of that farmer is in the database. |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal 1 Description: | | | |
|  | | | |
| ATTAINABLE Test | | | |
| What is your reaction to goal? | Does it feel realistic? | Is it Efficient | Do you find it motivating? |
| This is the most major or building block for the whole system | Yes, it is realistic because It requires moderate amount of efforts, more effective and hence won’t require high amount of time. | No, it is the basic necessity of the system. | The goal is motivating because it is the main part of the system. |

|  |  |  |
| --- | --- | --- |
| Goal 1 Description: | | |
| RELEVANT Test | | |
| Does it fit into the overall team / organization objective? | Taking overall fit is the timing appropriate? | Do you have sufficient resources / budget to succeed? |
| This forms the crucial part of the system and hence helps attain a problem that the Farmers faces. | Yes, Overall fit of the goal is the timing appropriate because for the Market process, details of the farmers is must. | To achieve the completion of this goal, the team will work on database system. |

|  |  |  |
| --- | --- | --- |
| Goal 1 Description: | | |
| TIME BOUND Test | | |
| Does it have a clear end date/point? | Is the focus clear so you can create an action plan? | Is its position on an Urgency/Importance grid clear? |
| Farmers can use the functions of this web-app if the details of that farmer is in the database. | The focus is to collect details of Farmers so that we can create profile of farmer and farmer can login to system and can use the functions of web-app | It is the basic requirement for the web-app. |

|  |  |  |
| --- | --- | --- |
| Goal-ID | Priority | Factors Addressed |
| 2 | 2 | Build Market agents (Customer’s) profile |
| Target Audience | Customers |
| Driver | To make Customer profile |
| Description | To populate whole database with Customer’s details. |
| Response | To extract customer’s data from database and performed operation on received information |
| Open Issues | Discussion and Revision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goal 2 Description: | | | | |
| Specific Test | | | | |
| Is ‘What’ identifiable? | Is the ‘Why’ clear? | Can ‘Who’ be identified? | ‘Where’ will it be performed? | ‘Which’ resources are needed? |
| Collecting data of Market Agents (Customers). | To populate the whole database with Customer’s details. | Customer will use data to login to this system and buy the foods of farmer. | Software engineer’s machines | Simple data entry and mongo dB atlas cloud for storing information. |

|  |  |  |
| --- | --- | --- |
| Goal 2 Description: | | |
| Measurable Test | | |
| Is the end result quantifiable? | ‘Figure’ of Measurement | Has the goal a clear end date/point? |
| Yes, as the Customers who have registered on the platform can be counted. | It can be measured on the basis of how many numbers of customers details are stored in the database | Customer must register on the web-app and provide details to buy the foods from the farmer |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal 2 Description: | | | |
| ATTAINABLE Test | | | |
| What is your reaction to goal? | Does it feel realistic? | Is it Efficient? | Do you find it motivating? |
| This will be one of the main goals to be achieved in this system | Yes, it is realistic because Customer registered through online web-app | Yes, it is since more customers using this module at a time hence that much maintenance is required | The goal is motivating because it is the main part of the system |

|  |  |  |
| --- | --- | --- |
| Goal 2 Description: | | |
| RELEVANT Test | | |
| Does it fit into the overall team / organization objective? | Taking overall fit is the timing appropriate? | Do you have sufficient resources / budget to succeed? |
| This forms the crucial part of the system and hence helps attain a problem that the organization faces. | Yes, Overall fit of the goal is the timing appropriate because for the Market process ,details of the Customer is must. | To achieve the completion of this goal, the team will works on database system. |

|  |  |  |
| --- | --- | --- |
| Goal 2 Description: | | |
| TIME BOUND Test | | |
| Does it have a clear end date/point? | Is the focus clear so you can create an action plan? | Is its position on an Urgency/Importance grid clear? |
| Customer must register on the web-app and provide details to buy the foods from the farmer | The focus of this goal is clear: to produce a Customer profile. So, steps forward can be taken. | Customer module is important to help the Farmers hence it is equally important. |

|  |  |  |
| --- | --- | --- |
| Goal-ID | Priority | Factors Addressed |
| 3 | 3 | Perform Add Crop production for selling |
| Target Audience | Farmers and Customers |
| Driver | To add crop production for selling to the market |
| Description | Farmers will add their crop production name, quantity and price for the crop and this advertisement of farmers crop will go for sell to Market |
| Response | The goal is to give best price for the farmers crop |
| Open Issues | Discussion and Revision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goal 3 Description: | | | | |
| Specific Test | | | | |
| Is ‘What’ identifiable? | Is the ‘Why’ clear? | Can ‘Who’ be identified? | ‘Where’ will it be performed? | ‘Which’ resources are needed? |
| Perform Add farmers crop production | For selling crop at good price to the Customers | Customers can see advertisement of crop available in Market added by the farmers | Software engineer’s machines | Simple data entry and mongodb atlas cloud for storing information. |

|  |  |  |
| --- | --- | --- |
| Goal 3 Description: | | |
| Measurable Test | | |
| Is the end result quantifiable? | ‘Figure’ of Measurement | Has the goal a clear end date/point? |
| It is a quantifiable result as adding crop in proper quantity can boost the market. | Added crops can be measured on the basis of unit of crop i.e. tonne, kgs etc. | The goal is to get good price for the farmer’s crop |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal 3 Description: | | | |
| ATTAINABLE Test | | | |
| What is your reaction to goal? | Does it feel realistic? | Is it Efficient? | Do you find it motivating? |
| This will be one of the main goals to be achieved in this system | Yes, it is realistic because farmers should add their crop production to get good price for crop | Yes, because if there are many orders then it is hard to maintain. | It is motivating because it adds value to the web-app. |

|  |  |  |
| --- | --- | --- |
| Goal 3 Description: | | |
| RELEVANT Test | | |
| Does it fit into the overall team / organization objective? | Taking overall fit is the timing appropriate? | Do you have sufficient resources / budget to succeed? |
| This forms the crucial part of the system that creates functionality to promote e-marketing for farmers crop production | Yes, Overall fit of the goal is the timing appropriate because for the Market process, adding crop production is must. | To achieve the completion of this goal, the team will work on database system. |

|  |  |  |
| --- | --- | --- |
| Goal 3 Description: | | |
| TIME BOUND TEST | | |
| Does it have a clear end date/point? | Is the focus clear so you can create an action plan? | Is its position on an Urgency/Importance grid clear? |
| The goal is to get the good price for the crops. | The focus of this goal is clear: to add a crop production So, steps forward can be taken. | It’s really important because if we don’t do that then web-app functionality will not fully be achieved. |

|  |  |  |
| --- | --- | --- |
| Goal-ID | Priority | Factors Addressed |
| 4 | 4 | Customer buy crop production |
| Target Audience | Customers |
| Driver | To buy crop production from farmer |
| Description | Customers will buy crop production added by the farmer.  After buying notification will go to the farmer and then farmer will accept or decline the order request. |
| Response | The goal is to focus on e-marketing module for farmers crop. |
| Open Issues | Discussion and Revision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goal 4 Description: | | | | |
| Specific Test | | | | |
| Is ‘What’ identifiable? | Is the ‘Why’ clear? | Can ‘Who’ be identified? | ‘Where’ will it be performed? | ‘Which’ resources are needed? |
| Perform buy farmers crop production | To make marketing of farmers crop effective and simpler. | Farmers can see which customers gives order and in how many quantity. | Software engineer’s machines | Simple data entry and mongodb atlas cloud for storing information. |

|  |  |  |
| --- | --- | --- |
| Goal 4 Description: | | |
| Measurable Test | | |
| Is the end result quantifiable? | ‘Figure’ of Measurement | Has the goal a clear end date/point? |
| It is a quantifiable result as farmers can receive sufficient orders. | It can be measured on the basis of number of orders received by the farmers. | The goal is to get sufficient orders for the farmers. |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal 4 Description: | | | |
| ATTAINABLE Test | | | |
| What is your reaction to goal? | Does it feel realistic? | Is it efficient? | Do you find it motivating? |
| This will be one of the main goals to be achieved in this system | Yes, it gives sense of e-marketing. | Yes, to provide sufficient orders to the farmers is sometime efficient. | It is motivating because it adds value to the web-app. |

|  |  |  |
| --- | --- | --- |
| Goal 4 Description: | | |
| RELEVANT Test | | |
| Does it fit into the overall team / organization objective? | Taking overall fit is the timing appropriate? | Do you have sufficient resources / budget to succeed? |
| This forms the crucial part of the system that creates functionality to promote e-marketing for farmers crop production | Yes, Overall fit of the goal is the timing appropriate because for the Market process ,buying crop production is must. | To achieve the completion of this goal, the team will works on database system. |

|  |  |  |
| --- | --- | --- |
| Goal 4 Description: | | |
| TIME BOUND Test | | |
| Does it have a clear end date/point? | Is the focus clear so you can create an action plan? | Is its position on an Urgency/Importance grid clear? |
| The goal is to get sufficient orders for the farmers. | The focus of this goal is clear: to buy a crop production So, steps forward can be taken. | It’s really important because if we don’t do that then web-app functionality will not fully be achieved. |

|  |  |  |
| --- | --- | --- |
| Goal-ID | Priority | Factors Addressed |
| 5 | 5 | Display weather |
| Target Audience | Farmers |
| Driver | To display 5-days weather to farmers |
| Description | Farmers can see weather of 5-days and take appropriate decision for their crops. |
| Response | The goal is to provide weather details to farmers so they can take good decisions. |
| Open Issues | Discussion and Revision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goal 5 Description: | | | | |
| Specific Test | | | | |
| Is ‘What’ identifiable? | Is the ‘Why’ clear? | Can ‘Who’ be identified? | ‘Where’ will it be performed? | ‘Which’ resources are needed? |
| Perform display weather for farmers | Farmers can see that weather and take decisions in Farming according to weather. | Farmers who are going to see weather. | Software engineer’s machines | Weather API’s are needed. |

|  |  |  |
| --- | --- | --- |
| Goal 5 Description: | | |
| Measurable Test | | |
| Is the end result quantifiable? | ‘Figure’ of Measurement | Has the goal a clear end date/point? |
| It is a quantifiable result as weather can be measured in degree Celsius. | Figure of Measurement is degree Celsius | The goal is to provide good UI of weather so that farmer can easily understand weather. |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal 5 Description: | | | |
| ATTAINABLE Test | | | |
| What is your reaction to goal? | Does it feel realistic? | Is it efficient? | Do you find it motivating? |
| It is achievable and also add extra feature to web app. | Yes, it is realistic because by using weather API we can achieve this. | The task plays an important role in the success of the project aim. | It is motivating because it will help the farmer. |

|  |  |  |
| --- | --- | --- |
| Goal 5 Description: | | |
| RELEVANT Test | | |
| Does it fit into the overall team / organization objective? | Taking overall fit is the timing appropriate? | Do you have sufficient resources / budget to succeed? |
| This forms the crucial part of the system that will help farmer  In their farming | Not take too much time it can be achieved within a given time. | Yes, weather API’s are available on the web. |

|  |  |  |
| --- | --- | --- |
| Goal 5 Description: | | |
| TIME BOUND Test | | |
| Does it have a clear end date/point? | Is the focus clear so you can create an action plan? | Is its position on an Urgency/Importance grid clear? |
| The goal is to provide good UI of weather so that farmer can easily understand weather. | The focus of this goal is clear: to show weather to farmers So, steps forward can be taken. | No because it is not main goal of app, It will just help farmers to see weather through this app. Farmers can also see weather from others website also. So it is not urgent. |

|  |  |  |
| --- | --- | --- |
| Goal-ID | Priority | Factors Addressed |
| 6 | 6 | Display Crop cycle |
| Target Audience | Farmers |
| Driver | To display crop cycle of crops selected by farmers. |
| Description | Farmers can see crop cycle i.e. (seeding->growth->Harvesting) |
| Response | The goal is to provide crop cycle to farmer so that they can get guidance in their farming. |
| Open Issues | Discussion and Revision |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Goal 5 Description: | | | | |
| Specific Test | | | | |
| Is ‘What’ identifiable? | Is the ‘Why’ clear? | Can ‘Who’ be identified? | ‘Where’ will it be performed? | ‘Which’ resources are needed? |
| Perform display Crop cycle for farmers | Farmers can see crop cycle and take help from this in their farming. | Farmers who are going to see crop cycle. | Software engineer’s machines | Crop information is needed from seeding to harvesting . |

|  |  |  |
| --- | --- | --- |
| Goal 5 Description: | | |
| Measurable Test | | |
| Is the end result quantifiable? | ‘Figure’ of Measurement | Has the goal a clear end date/point? |
| Sufficient amount of information is available of each crop in crop-cycle. | Number of crops are available in crop-cycle added by the farmer. | The goal completion will go hand in hand with the project completion. |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal 5 Description: | | | |
| ATTAINABLE Test | | | |
| What is your reaction to goal? | Does it feel realistic? | Is it efficient? | Do you find it motivating? |
| It is achievable and also add extra feature to web app. | Yes, it is realistic because by information of crop is available on the web. | The task plays an important role in the success of the project aim. | It is motivating because it will help the farmer. |

|  |  |  |
| --- | --- | --- |
| Goal 5 Description: | | |
| RELEVANT Test | | |
| Does it fit into the overall team / organization objective? | Taking overall fit is the timing appropriate? | Do you have sufficient resources / budget to succeed? |
| Yes, it will enhance app features and also helps the farmers. | It requires some more time for actual implementation. | Yes, information about all steps of crops in farming are available on the web. |

|  |  |  |
| --- | --- | --- |
| Goal 5 Description: | | |
| TIME BOUND Test | | |
| Does it have a clear end date/point? | Is the focus clear so you can create an action plan? | Is its position on an Urgency/Importance grid clear? |
| The goal completion will go hand in hand with the project completion. | The focus of this goal is clear: to provide good and sufficient information to the farmer about their crops. | No because it is not main goal of app, It will just help farmers to see weather through this app. Farmers can also see information on other apps and on web.so it is not urgent. |

# 5. FEATURE SET

*These are the features that make our product unique.*

|  |  |
| --- | --- |
| Feature-ID | Feature Description |
| 1 | Provides report of daily sales summary. |
| 2 | Gives details about every farmers sale. |
| 3 | Purchase or order details. |
| 4 | Details about customer and their records. |
| 5 | Report of customer’s item request and item information. |
| 6 | Gives details of weather to farmer. |
| 7 | Details of Crop-Cycle of crops selected by farmer. |

# 6. STAKEHOLDERS

|  |  |  |  |
| --- | --- | --- | --- |
| Stakeholder | Concerns | Quadrant | Strategy/ Benefits |
| Project Guide | Ensuring proper handover of project to operations team | Minimal Effort | Communicate project specifications as required |
|  | Resource and scheduling constraints for production once project is transitioned to operations | Key Player | Solicit stakeholder as member of steering committee and obtain feedback on project planning. Frequent communication and addressing concerns are imperative |
| Leader | Ensuring on time delivery of materials | Minimal Effort | Communicate project schedule and material requirements ahead of time to ensure delivery |
| Developer | Product performance must meet or exceed current product | Keep Informed | Communicate test results and performance specifications and obtain feedback on customer requirements or any changes. Provide frequent status reports and updates. |
| F | Concerns regarding resources to assist project team with product design | Keep Satisfied | Communicate applicable resource requirements early and ensure resources are released back to engineering when they’re no longer required |
| Designer | Questions regarding design of product | Keep Informed | Allow technical staff to work with stakeholder to answer questions and address concerns and provide test results for validation |

# 7. ACCEPTANCE CRITERIA

This project as a whole, including the technical solution, will be deemed a success if:

1. The project is delivered on schedule.
2. System is stable and the design is as per the current standards and should use recommended coding practices.
3. All objective laid above should be fulfilled by the system.

|  |  |  |
| --- | --- | --- |
| Item | Concerns | Accepted / Rejected |
| Vision Definition | **Complexity** | **Accepted** |
| Mission Definition | **Relation with Deliverables** | **Accepted** |
| Goals | **Description and structure** | **Accepted** |
| Feature Definitions | **Readability for Farmers and Customers** | **Accepted** |
| Deliverables definition | **Consistency** | **Accepted** |

**T.Y. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 3

**ERP-Farmers**

**System Requirement Specification**

***15-03-2019***

***Version 1.0***

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front End** |
| 62 | **161259** | **Aniket Thaware** | **Backend** |
| 73 | **161854** | **Rushabh pahade** | **Leader** |

**Approved By: Dr M. R. Dube**

**Academic Year: 2018-19 Semester: II**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction | **23** |
| 2 | Terms of Reference | **24** |
| 3 | Problem Description | **24** |
| 4 | Functional Hierarchy | **25** |
| 5 | User Interfaces | **26** |
| 6 | Hardware Interfaces | **26** |
| 7 | Software Interfaces | **27** |
| 8 | Logical Databases | **28** |
| 9 | Non-Functional Requirements | **30** |

# 1. INTRODUCTION

*.*

The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. The aim of this document is to gather and analyze and give an in-depth insight of the complete **ERP farmers** software system by defining the problem statement in detail. Nevertheless, it also concentrates on the capabilities required by stakeholders and their needs while defining high-level product features.

|  |  |
| --- | --- |
| Item | Description |
| Purpose | **To get the system requirements of the farmers. The purpose of the project is to help farmers get more profit, using technology more farmers can be connected to the market.** |
| Audiences | **Farmers, customers and admin** |
| SRS Scope | **This SRS is aimed at specifying requirements of software to be developed. This standard can be used to create software requirements specifications or project specific standard. It does not identify any specific method or tool for preparing an SRS.** |
| Project Scope | **The main scope is to increase market value and help farmers to get more profit.** |

# 2. TERMS OF REFERENCE

|  |  |
| --- | --- |
| 1. **Background** | 1. To make farmers work easy. 2. In ‘ERP-Farmers’, authorised users are Farmers and Customers. 3. Area of working is Mongo-db, coding, designing and testing. 4. ‘ERP-Farmers’ can reach out to more Farmers to find customer for their crops. |
| 1. **Objectives** | 1. The objective of ‘ERP-Farmers’ is to provide perfect platform for Farmers to sell their crops , see weather conditions and information about the crops. 2. By this proposal, Farmers and Customers can save their time easily. 3. To save the unnecessary expenditure of the customer |
| 1. **Issues** | 1. Relevance – Proving relevance to stakeholders. 2. Effectiveness – Exact outputs of the project and realization of benefits. 3. Impact – the market is ever growing and the product’s use will never cease to exist. 4. Sustainability – The project if successful will have wide ranging benefits and will become self-sustainable soon after its initiation. 5. Efficiency- Efficiency of the algorithm and its outcome to predict values. |
| 1. **Methodology** | 1. For completion of the work under the resulting contract, the work of system will be subdivided in the team members of the team. Designing of the system by using web language create a front end of the system. 2. After that data collection, data analysis as well as validation of that data will take place so that it can be easily handle by company’s administration. 3. Designing of integrate system and the back end of the system will be formed simultaneously with the data operations. 4. Immediately after, the preview of the system will show to the company and their feedback about the system will take in considerations. 5. At the final stage, the delivery of proposed system will take place. |
| 1. **Expertise** | 1. The type of work involved in the system is easily accessible UI development. 2. The type of skills and abilities required for the system is to know the knowledge about database, web designing language, etc. 3. The 3 customers from TY\_C is involved in the development of the system 4. The period of engagement of each team member is roughly 4 to 5month |
| 1. **Reporting** | 1. Reports inform time to time progress of the system to the Guide. 2. The timely report will give to the company as per the schedule of the deliverables. 3. The computer software program’s to be used for report writing is Microsoft word. 4. The people responsible for reporting and approving are the members of the team and authority of the company. 5. The timely report card will give to the leader, authority of the company and managing director of the company. |
| 1. **Work plan** | 1. The anticipated work will give with the report card. 2. The finance resources allocated to the system will divide in the 3 categories in the form of the cash. One will be at the initial stage, 2nd will be in the middle one and the last payment after the delivery of the system. |

# 3. PROBLEM DESCRIPTION

|  |  |
| --- | --- |
| The problem of | Low market value for crops and less market exposure. |
| Affects | Farmer and related people |
| The impact of which is | 1. The farmer will be exposed to wide area of market to sell its crops.  2. Customers from different regions will be able to interact directly to the farmers, it will reduce the price of crops for customers and indirectly increase its profit.  3. The farmer will also be profited by selling the crops. |
| A successful solution would | Benefit both the farmer and the customers |

|  |  |
| --- | --- |
| For | Customers (End user) |
| Who | To get the write database and we can excel the profit by large margin |
| The ERP software | is a software solution |
| That | Helps farmers to achieve high income |
| Unlike | Traditional approach |
| Our product | Is practical |

# 4. FUNCTIONAL HIERARCHY

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 1 | Goal Name | Description |
| Objective ID | 1 | To get required data | Contacting various agriculture colleges for farmer database. |
| Process ID: 1 | Contacting farmers for data |
| Process ID: 2 | Collecting and analysing data |
| Objective ID | 2 | Storing data | Storing data in database |
| Process ID: 1 | Converting data into particular format |
| Process ID: 2 | Storing the data |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 2 | Goal Name | Description |
| Objective ID | 1 | Market analysis | Analysing market |
| Process ID: 1 | Getting market information |
| Process ID: 2 | Seeking market information from market database |
| Objective ID | 2 | Filtering market data | Filtering of data |
| Process ID: 1 | Checking the data |
| Process ID: 2 | If its correct then posting on the website |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 3 | Goal Name | Description |
| Objective ID | 1 | Connecting farmer with the market | The most important thing is to connect the users |
| Process ID: 1 | The ad of farmer must be posted in market |
| Process ID: 2 | Now the customers can buy the goods from farmers |
| Objective ID | 2 | Status of orders | Order status |
| Process ID: 1 | The status of the order is available |
| Process ID: 2 | Orders must be executed |

|  |  |  |  |
| --- | --- | --- | --- |
| Goal-ID | 4 | Goal Name | Description |
| Objective ID | 1 | Objective Name | Get feedback |
| Process ID: 1 | Gather feedback from customers |
| Process ID: 2 | Store feedback |
| Objective ID | 2 | Objective Name | Understanding feedback |
| Process ID: 1 | Work on the feedback |
| Process ID: 2 | Send to higher authorities to improve the system |
|  |  |  |  |

# 5. USER INTERFACES

5.1 Abbreviated UI, it is the junction between a user and a computer program. An interface is a set of commands or menus through which a user communicates with a program. A command-driven interface is one in which you enter commands. A menu-driven interface is one in which you select command choices from various menus displayed on the screen.

The user interface is one of the most important parts of any program because it determines how easily you can make the program do what you want. A powerful program with a poorly designed user interface has little value. Graphical user interfaces (GUIs) that use windows, icons, and pop-up menus have become standard on personal computers.

GUI is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.

Graphical user interfaces, such as Microsoft Windows and the one used by the Apple Macintosh, feature the following basic components:

* Pointer: A symbol that appears on the display screen and that you move to select objects and commands. Usually, the pointer appears as a small angled arrow. Text -processing applications, however, use an I-beam pointer that is shaped like a capital I.
* Pointing device: A device, such as a mouse or trackball, that enables you to select objects on the display screen.
* Icons: Small pictures that represent commands, files, or windows. By moving the pointer to the icon and pressing a mouse button, you can execute a command or convert the icon into a window. You can also move the icons around the display screen as if they were real objects on your desk.
* Desktop: The area on the display screen where icons are grouped is often referred to as the desktop because the icons are intended to represent real objects on a real desktop.
* Windows: You can divide the screen into different areas. In each window, you can run a different program or display a different file. You can move windows around the display screen, and change their shape and size at will.
* Menus: Most graphical user interfaces let you execute commands by selecting a choice from a menu.

In addition to their visual components, graphical user interfaces also make it easier to move data from one application to another. A true GUI includes standard formats for representing text and graphics. Because the formats are well-defined, different programs that run under a common GUI can share data. This makes it possible, for example, to copy a graph created by a spreadsheet program into a document created by a word processor.

5.2 Characteristics of Successful User Interfaces

* **Clear**: Clarity is the most important element of user interface design. Indeed, the whole purpose of user interface design is to enable people to interact with your system by communicating meaning and function. If people can’t figure out how your application works or where to go on your website they’ll get confused and frustrated.
* **Concise**: Clarity in a user interface is great, however, you should be careful not to fall into the trap of over-clarifying. It is easy to add definitions and explanations, but every time you do that you add mass. Your interface grows. Add too many explanations and your users will have to spend too much time reading through them. Keep things clear but also keep things concise. When you can explain a feature in one sentence instead of three, do it. When you can label an item with one word instead of two, do it. Save the valuable time of your users by keeping things concise. Keeping things clear and concise at the same time isn’t easy and takes time and effort to achieve, but the rewards are great.
* **Familiar**: Many designers strive to make their interfaces ‘intuitive’. But what does intuitive really mean? It means something that can be naturally and instinctively understood and comprehended. But how can you make something intuitive? You do it by making it ‘familiar’. Familiar is just that: something which appears like something else you’ve encountered before. When you’re familiar with something, you know how it behaves – you know what to expect. Identify things that are familiar to your users and integrate them into your user interface.
* **Responsive**: Responsive means a couple of things. First of all, responsive means fast. The interface, if not the software behind it, should work fast. Waiting for things to load and using slaggy and slow interfaces is frustrating. Seeing things load quickly, or at the very least, an interface that loads quickly (even if the content is yet to catch up) improves the user experience. Responsive also means the interface provides some form of feedback. The interface should talk back to the user to inform them about what’s happening. Have you pressed that button successfully? How would you know? The button should display a ‘pressed’ state to give that feedback.
* **Consistent**: Consistent interfaces allow users to develop usage patterns – they’ll learn what the different buttons, tabs, icons and other interface elements look like and will recognize them and realize what they do in different contexts. They’ll also learn how certain things work, and will be able to work out how to operate new features quicker, extrapolating from those previous experiences.
* **Attractive**: This one may be a little controversial but I believe a good interface should be attractive. Attractive in a sense that it makes the use of that interface enjoyable. Yes, you can make your UI simple, easy to use, efficient and responsive, and it will do its job well – but if you can go that extra step further and make it attractive, then you will make the experience of using that interface truly satisfying. When your software is pleasant to use, your customers or staff will not simply be using it – they’ll look forward to using it. There are of course many different types of software and websites, all produced for different markets and audiences. What looks ‘good’ for any one particular audience will vary. This means that you should fashion the look and feel of your interface for your audience. Also, aesthetics should be used in moderation and to reinforce function. Adding a level of polish to the interface is different to loading it with superfluous eye-candy.
* **Efficient**: A user interface is the vehicle that takes you places. Those places are the different functions of the software application or website. A good interface should allow you to perform those functions faster and with less effort. Now, ‘efficient’ sounds like a fairly vague attribute – if you combine all of the other things on this list, surely the interface will end up being efficient? Almost, but not quite. What you really need to do to make an interface efficient is to figure out what exactly the user is trying to achieve, and then let them do exactly that without any fuss. You have to identify how your application should ‘work’ – what functions does it need to have, what are the goals you’re trying to achieve? Implement an interface that lets people easily accomplish what they want instead of simply implementing access to a list of features.
* **Forgiving**: Nobody is perfect, and people are bound to make mistakes when using your software or website. How well you can handle those mistakes will be an important indicator of your software’s quality. Don’t punish the user – build a forgiving interface to remedy issues that come up. A forgiving interface is one that can save your users from costly mistakes. For example, if someone deletes an important piece of information, can they easily retrieve it or undo this action? When someone navigates to a broken or non-existent page on your website, what do they see? Are they greeted with a cryptic error or do they get a helpful list of alternative destinations?

|  |  |  |  |
| --- | --- | --- | --- |
| UI-ID | UI Name | Type | Scope |
| 1 | **Register Farmer** | **Input** | **Farmers are added to the database** |
| 2 | **Register Customer** | **Input** | **Customers are added to the database** |
| 3 | **Main Page** | **Navigation** | **Display All Features of the Web-Application** |
| 4 | **Add crop** | **Forms** | **Adding crops that can be sold** |
| 5 | **Market** | **Navigation** | **The goods to be sold are send to market** |
| 6 | **Analysing market** | **Command** | **The crops are filtered according to the crop type** |
| 7 | **My cart** | **Menu** | **The crops are delivered effectively** |
| 8 | **Feedback** | **Input** | **The feedback is used to improve the system.** |
| 9 | **Weather** | **Output** | **Display the weather information** |
| 10 | **Login Farmer** | **Input** | **Registered Farmers are given access to the database** |
| 11 | **Login Customer** | **Input** | **Registered Customers are given access to market database** |
| 12 | **Admin Login Page** | **Input** | **Admin is given access to modify the database** |
| 13 | **Farmer Profile Page** | **Output** | **Display the information about farmer from the database** |
| 14 | **Farmer Profile Edit Page** | **Input** | **Edit the information of the farmer and update it to the database** |
| 15 | **Customer Profile Page** | **Output** | **Display the information about Customer from the database** |
| 16 | **Customer Profile Edit Page** | **Input** | **Edit the information of the customer and update It to the database** |
| 17 | **Local Agriculture News Page** | **Output** | **Display the news of the local agriculture market near the farmer** |
| 18 | **Dealers Page** | **Output** | **Display the information about the dealers nearby the farmer.** |
| 19 | **Display Advertisements Page** | **Output** | **Display the Advertisements posted by the farmers to the customers** |
| 20 | **Add Advertisement Page** | **Input** | **Add the information to post the advertisement for the customers** |
| 21 | **Governments Schemes Page** | **Output** | **Display the various schemes provided by the Government for the farmers** |
| 22 | **Helpline Page** | **Output** | **This page will be guide to the farmers** |
| 23 | **Dealers Page** | **Output** | **Display the information about the dealers nearby the farmer** |
| 24 | **About Page** | **Output** | **Display the information about the web application** |

# 6. HARDWARE INTERFACES

|  |  |
| --- | --- |
| Profile | Description |
| Processor | **Intel 7th gen or higher** |
| RAM | **8 GB** |
| Server Side Technology | * Database storage space: 512 MB * Monitor of resolution 1024 x 768 |
| Client Side Technology | * Monitor of resolution 1024 x 768 * Working Internet Connection and Port |
| External Devices | * Monitor * Mouse * Keyboard |

# 7. SOFTWARE INTERFACES

|  |  |
| --- | --- |
| Profile | Description |
| Front-end Capabilities | **Browser, HTML5 support** |
| Back-end Capabilities | **MongoDB** |
| Programming Languages | **Java, HTML** |
| Operating Environment | **NA** |
| Software Platform | **Browser** |
| Database Servers | **MongoDB** |
| Framework Resources | **Grails** |
| API (If Any) | **Weather API** |
| Other Services/Resources | **NA** |
| Communication Interfaces | **Internet** |

# 8. LOGICAL DATABASES

|  |  |  |
| --- | --- | --- |
| Database Name | Parameter | Scope |
| Farmer database | **All the basic details of farmers** | **Input data** |
| Market database | **All the basic details of market** | **Input data** |
| Crop database | **All the details of crops** | **Input data** |
| Ads database | **The ads posted by farmers will be stored** | **Input/output data** |
| Market requirements | **The price and demand for each crop** | **Input/output data** |
| Orders | **The finished orders will be stored** | **Input/output data** |

# 9. NON-FUNCTIONAL REQUIREMENTS

* Reliability: Specify the factors required to establish the required reliability of the software system at time of delivery. If you have MTBF requirements, express them here. This doesn’t refer to just having a program that does not crash. This has a specific engineering meaning.
* Availability: Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. This is somewhat related to reliability. Some systems run only infrequently on-demand (like MS Word). Some systems have to run 24/7 (like an e-commerce web site). The required availability will greatly impact the design. What are the requirements for system recovery from a failure? “The system shall allow users to restart the application after failure with the loss of at most 12 characters of input”.
* Security: Specify the factors that would protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:
  + Utilize certain cryptographic techniques
  + Keep specific log or history data sets
  + Assign certain functions to different modules
  + Restrict communications between some areas of the program
  + Check data integrity for critical variables
* Maintainability: Specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices. If someone else will maintain the system
* Portability: Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include:
  + Percentage of components with host-dependent code
  + Percentage of code that is host dependent
  + Use of a proven portable language
  + Use of a particular compiler or language subset
  + Use of a particular operating system
* Correctness - extent to which program satisfies specifications, fulfills user’s mission objectives
* Efficiency - amount of computing resources and code required to perform function
* Flexibility - effort needed to modify operational program
* Interoperability - effort needed to couple one system with another
* Reliability - extent to which program performs with required precision
* Reusability - extent to which it can be reused in another application
* Testability - effort needed to test to ensure performs as intended
* Usability - effort required to learn, operate, prepare input, and interpret output

Once the relevant characteristics are selected, a subsection should be written for each, explaining the rationale for including this characteristic and how it will be tested and measured. A chart like this might be used to identify the key characteristics (rating them High or Medium), then identifying which are preferred when trading off design or implementation decisions (with the ID of the preferred one indicated in the chart to the right). The chart below is optional (it can be confusing) and is for demonstrating trade-off analysis between different non-functional requirements. H/M/L is the relative priority of that non-functional requirement.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Characteristic** | **H/M/L** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| 1 | Correctness | H |  | 2 |  |  |  |  |  |  |  |  |  |  |
| 2 | Efficiency | L |  |  |  |  |  |  |  |  |  | 10 |  |  |
| 3 | Flexibility | L |  |  |  |  |  |  |  |  |  |  | 11 |  |
| 4 | Integrity/Security | H |  |  |  |  |  | 6 |  |  |  |  |  |  |
| 5 | Interoperability | M |  |  |  |  |  |  |  |  | 9 |  |  |  |
| 6 | Maintainability | H |  |  |  |  | 5 |  |  |  |  |  |  |  |
| 7 | Portability | M |  |  |  |  |  |  | 7 |  |  |  |  |  |
| 8 | Reliability | H | 1 |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Reusability | L |  |  |  |  |  |  |  |  |  |  |  | 12 |
| 10 | Testability | M |  |  |  |  |  |  |  | 8 |  |  |  |  |
| 11 | Usability | H |  |  | 3 |  |  |  |  |  |  |  |  |  |
| 12 | Availability | M |  |  |  | 4 |  |  |  |  |  |  |  |  |

**T.Y. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 4

**ERP-Farmers**

**Feasibility Study Report**

***22-03-2019***

***VERSION 1.0***

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front End** |
| 62 | **161259** | **Aniket Thaware** | **Backend** |
| 73 | **161854** | **Rushabh pahade** | **Leader** |

**Approved By: Dr M. R. Dube**

**Academic Year: 2018-19 Semester: II**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction | **33** |
| 2 | Description of Services | **34** |
| 3 | Technology Considerations | **34** |
| 4 | Feasibility Study Results | **35** |
| 5 | References | **36** |

# 1. INTRODUCTION

|  |  |
| --- | --- |
| Item | Description |
| Scope of Study | 1.Through this project we want to connect more customers to the farmers in order to increase farmer’s income.  2.To give farmer a good estimate of the price of the crops. |
| Audiences | 1.Farmers  2.Market agents looking to buy crops |
| Project Type | Medium Scale |
| Platform Details | Existing technologies :  Front End:  1.Html  2.CSS  3.PHP  Back End:  1.MongoDB  2.Grails  3.Groovy language |

# 2. DESCRIPTION OF SERVICES

|  |  |  |  |
| --- | --- | --- | --- |
| Service -ID | Service Name | Audience | Scope |
| S-1 | Collect farmer data | Stakeholder | Detailed information about farmer |
| S-2 | Storing the data in database | User | Storing the farmer data in farmer database |
| S-3 | Display farmer information | End User | The user will see the total information about farmer |
| S-4 | Identify the market | Stakeholder | Getting information of the nearest market |
| S-5 | Connecting famers with market | End user | The data from farmers will be posted as ads in market, so market can see the ads from all the farmers. |
| S-6 | Display weather information | End user | The famer will be able to see the weather information. |
| S-7 | Examine Feedback | End user | Feedback will be considered and processed. |

# 3. TECHNOLOGY CONSIDERATIONS

|  |  |  |
| --- | --- | --- |
| Current Technology | | |
| Type | **Parameter** | **Description** |
| Hardware | CPU | Intel core i5 7th generation, 2.5 GHz with turbo boost Upto 2.7 Ghz. |
|  | RAM | 8GB |
|  | GPU | Geforce 940MX |
| Software | IDE’s | Intellij, Brackets |
|  | Server | MongoDB |
|  | Browser | Chrome |

|  |  |  |
| --- | --- | --- |
| Deployment Technology | | |
| Type | **Parameter** | **Description** |
| Hardware | Device | Desktop, laptop, mobile |
|  | Screen | Screen with minimum 1024x576 resolution |
| Software | Browser | Chrome, Mozilla |
|  | Support | Html5, CSS, BOOTSTARP, JavaScript |

# 4. FEASIBILITY STUDY RESULTS

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Outcome | Ranking | Discussion |
| Collect Farmer Information | Expected | H | The Farmer Information Data is available and is successfully retrieved. |
| Unexpected | L | The Farmer Information is not available due to lack of communication. |
| Check Farmer Database Correctness | Expected | H | The Database collected is correct and is in required Format. |
| Unexpected | L | The Database is not available in precise Format. If the Data (Passwords or Usernames) is missing or incorrect it can be replaced by a certain value. If the Data is not available for a particular Farmer then new Data must be inserted. |
| Display Farmer Information | Expected | H | Required farmer information is available and display correctly. |
| Unexpected | L | Farmer information is unavailable and displaying is not possible. |
| Process feedback | Expected | M | Feedback is positive. The value generated are precise and consistent. |
| Unexpected | M | Feedback is negative. |

# 5. REFERENCES

*1.* [*https://www.w3schools.com//*](https://www.w3schools.com//)

*2. https://docs.grails.org/3.3.9/guide/single.html*

*3. Agriculture college database*

*4. Farmer.gov.in*

**T.Y. B. Tech.**

**CS 3001: Software Engineering Laboratory**

Assignment No: 5

**ERP-Farmers**

**Project Plan Outline**

***28-03-2019***

***Version 1.0***

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front End** |
| 62 | **161259** | **Aniket Thaware** | **Backend** |
| 73 | **161854** | **Rushabh pahade** | **Leader** |

**Approved By: Dr M. R. Dube**

**Academic Year: 2018-19 Semester: II**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction | **37** |
| 2 | Project Milestones | **37** |
| 3 | Work Breakdown Structure | **38** |
| 4 | Project Communication | **39** |
| 5 | Activity Register | **39** |
| 6 | Task Prioritization | **40** |
| 7 | Risk Register | **41** |

# INTRODUCTION

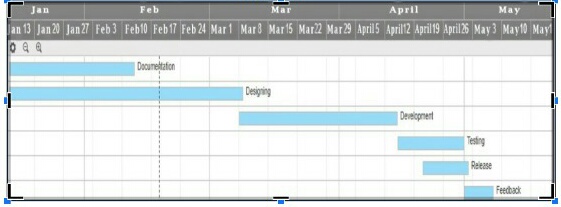
|  |  |
| --- | --- |
| Deliverables | Benefits |
| 1. SOW | Gives an idea of what the system is. |
| 2. Feature Set | Provides the set of features the system will provide. |
| 3. SRS | Specifies the requirements for the system. |
| 4. Feasibility Study | Gives an account of how feasible it is to use the system. |
| 5. Project Plan | Will provide information on how the project will be executed. |
| 6. Sprint Level Planning Activity | Planning will help in easy execution of the system. |
| 7. Sprint Level Design Activity | Preparing the design will make the implementation faster because a blueprint will be available. |
| 8. Software Configuration Management Plan | It will make the execution of the software much easier as there is a plan in place. |
| 9. Sprint Execution | The system will be available to use as early as possible. |
| 10. Sprint Review | Fast review of the system so that so that errors can be removed as early as possible. |

# PROJECT MILESTONES

|  |  |  |
| --- | --- | --- |
| Milestones | Phase | Description |
| 1 | Inception | Delivering Statement of Work document |
| 2 | Inception | Delivering Feature Set document |
| 3 | Elaboration | Feasibility study and Project Plan using AGILE |
| 4 | Elaboration | Sprint level planning activity |
| 5 | Construction | Sprint Plan and Sprint Design |
| 6 | Construction | Software Configuration Management Plan (SCMP) and Sprint Execution |
| 7 | Transition | Sprint Review and Sign- offs |
| 8 | Transition | Feedback |

# WORK BREAKDOWN STRUCTURE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WBS ID | WBS Package | Role | Description | Delivery Date |
| 1 | Documentation | Inception | Creation of SOW, FRS, SRS | 12 Sept 2017 |
| 2 | Designing | Elaboration | Making Prototypes | 08 Oct 2017 |
| 3 | Development | Construction | Development of Real System using appropriate languages | 14 Nov 2017 |
| 4 | Testing | Construction | Testing of System for Defects and checking for correctness | 30 Nov 2017 |
| 5 | Product Release | Transition | Marketing, Managing of the System in live environment | 1 Dec 2017 |
| 6 | Feedback | Transition | Taking user experience as feedback and modifying System | 7 Dec 2017 |

**GANTT CHART******

# PROJECT COMMUNICATION

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Communication Type | Description | Frequency | Format | Participants/ Distribution | Deliverable | Owner |
| Weekly Status Report | Email summary of project status | Weekly | In Person | Project Guide,  Project Team | Status Report | Project Manager |
| Weekly Project Team Meeting | Meeting to review action register and status | Weekly | In Person | Project Team | Updated Action Register | Project Manager |
| Project Monthly Review (PMR) | Present metrics and status to team and sponsor | As Needed | In Person | Project Guide, Team, and Stakeholders | Status and Metric Presentation | Project Manager |
| Project Gate Reviews | Present closeout of project phases and kick-off next phase | As Needed | In Person | Project Sponsor, Team and Stakeholders | Phase completion report and phase kick-off | Project Manager |
| Technical Design Review | Review of any technical designs or work associated with the project | As Needed | In Person | Project Team | Technical Design Package | Project Manager |

# ACTIVITY REGISTER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Activity Number** | **Activity Name** | **Activity description** | **Responsibility** | **Comments** |
| 1 | Prepare  Documentation | * Create Project Initiation Documents | * Rohan Bhukne is responsible for coordinating with the team. | * Meet Deadlines |
| * Documents: SOW, Feature Set and SRS | * WBS Package 1 |
| 2 | Conceptualise Design | * Evaluate Feasibility | * Rohan Bhukne is responsible for execution of project planning phase. | * Quick Execution Required |
| * Develop Project Plan | * WBS Package 2 |
| 3 | Collect Data | * Acquire Data from Sources on the Internet | * Aniket Thaware is responsible for acquiring correct data | * WBS Package 2 |
| * Important phase for smoot development |
| 4 | Developing System | * Develop Machine Learning Model | * Aniket Thaware is responsible for delegating everyone with instructions for development. | * Development in Sprints |
| * Implement Model to Predict Values | * WBS Package 3 |
| 5 | Design UI | * Create User Interface | * Rohan Bhukne will oversee the UI creation activity. | * WBS package3 |
| * Design UI to appropriately display the statistics | * The phase execution will have to run parallelly with development stage |
| 6 | Checking for bugs | * Unit and System Testing | * Rushabh Pahade will be in charge of creating test cases and checking for bugs | * Preparing Test Cases * WBS Package 4 |
| * Debugging |
| 7 | Releasing Product | * Advertising System | * Rushabh Pahade will be responsible for the marketing of the product. * Rushabh Pahade will also share the responsibility. | * Good Marketing Strategies * WBS Package 4 |
| * Finding Clients |
| 8 | Feedback of System | * Taking reviews from customers * Implementing new features | * Aniket Thaware will oversee the feedback and update activities. | * Understanding what changes are needed * WBS Package 6 |

# 6. TASKS PRIORITAZATION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Task is of high importance, with high urgency factor.***  *Must be done today & to high standard.*  *Action ASAP* |  | ***High Importance*** | ***Low Importance*** | ***Task is of low importance, with high urgency factor.***  *These tasks need to be completed on time.*  *ONLY spend sufficient time on them as not important.*  *Don’t be diverted* |
| ***High Urgency*** | ***1.Collect farmers data***  ***2. Collect crops data***  ***3. Initial documentation*** | ***1. Display Weather***  ***2. Crop cycle*** |
| ***Task is of high importance, but has low urgency factor.***  *By nature long-term so need to:*   1. *Set target if none exists.* 2. *Break-up into chunks of work* | ***Low Urgency*** | ***1. Create user interface***  ***2. Determine system accuracy***  ***3. Version control mechanism*** | ***1. Update the database on user feedback*** | ***Task is both low in importance & urgency.***  *Discard as many of these tasks as possible because they cause great harm to your productivity.*  *Delegate if they develop another’s KSA’s.* |

# 7. RISK REGISTER

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Risk Description** | **Likely Cause of Risk Occurring** | **Effect on Project** | **Phase Affected** | ***Severity Level*** | **Ability to Detect** | **Risk Rank** |
| **1** | **farmer’s data not available** | **1.NOT ENOUGH DATA AVAILABLE**  **2. data not available for open use** | **failure to create farmer’s profile** | **inception** | ***High*** | **Moderate** | **Serious** |
| **2.** | **farmer’s data is incorrect** | **data source might have anomalies** | **details need to be fetched again** | **testing** | ***Med*** | **Moderate** | **Serious** |
| **3** | **data collected is outdated** | **no use of such data** | **project can’t be proceed** | **elaboration** | ***Med*** | **Moderate** | **Critical** |
| **4** | **insufficient data** | **data available is insufficient** | **for generalization** |  | ***High***  ***Med***  ***Low*** | **Easy**  **Moderate**  **Complex** | **Critical**  **Serious**  **Modest**  **Trivial** |

**T.Y. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 6

**ERP-farmers**

**Project Backlog**

***Version 1.0***

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front End** |
| 62 | **161259** | **Aniket Thaware** | **Backend** |
| 73 | **161854** | **Rushabh pahade** | **Leader** |

**Approved By:**

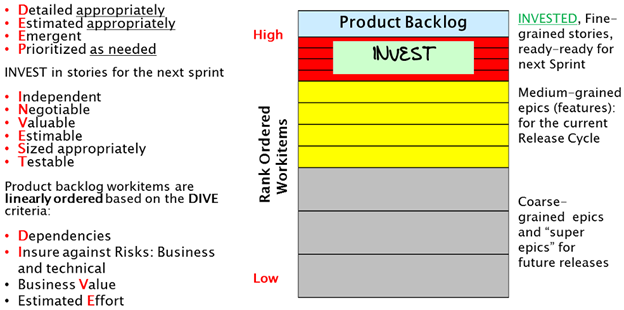
**Academic Year: 2018-19 Semester: II**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction | **44** |
| 2 | Sprint Planning And Work-Item Granularity | **45** |
| 3 | Release Planning And Work Granularity | **46** |
| 4 | Product Planning And Work-Item Granularity | **48** |
| 5 | Product Backlog: Goals Granularity | **50** |

# 1. INTRODUCTION

A product backlog stores, organizes and manages all work items that you plan to work on in the future. The key characteristics of a well-organized and managed product backlog are summarized in the image below. DEEP, INVEST and DIVE are meaningful words.



*Figure 1: Characteristics of a Managed Product Backlog*

The **granularity** or size of work items should be determined based on how far into the future you are planning a product, i.e., the planning horizon. It is the observation that the longer or shorter the planning horizon, the larger or smaller the work items. This makes sense as it takes a lot more effort to develop, specify and maintain a large number of small-grain work items compared to developing, specifying and maintaining a small number of large-grain work items. Smaller work items, stories, are typically developed by breaking down larger work items, epics. Stories are the unit of software design, development and value delivery.

**DEEP product backlog**

A product backlog may have several hundred or more work items, hence the acronym DEEP. Work items can be comprised of stories, defects and test sets. DEEP is acronym capturing the essence of the logical structure of product backlog.

* **Detailed appropriately**: Work-items in the backlog are specified at an appropriate level of detail.
* **Estimated appropriately**: Work-items in the product backlog are estimated appropriately.
* **Emergent**: Product backlog is not frozen or static; it evolves or emerges on an on-going basis in response to product feedback, and changes in competitive, market and business. New backlog items are added, existing items are groomed (revised, refined, elaborated) or deleted or re-prioritized.
* **Prioritized as needed**: Work-items in the backlog are linearly rank-ordered as needed.

# 2. SPRINT PLANNING AND WORK-ITEM GRANURALITY

If the planning horizon is the next, i.e., upcoming sprint or iteration (typically 2 to 4 weeks), each Work-items is small enough to fit in a single sprint, and is 100% ready (“ready-ready”) to be worked on, as indicated in Figure 1 – see the top red-colour region. A ready-ready story has already been analysed with clear definition (User Role, Functionality, and Business Value) and associated Acceptance Criteria. Work-items planned for the next sprint are stories, defects and test sets. The Work-items in the next sprint have the highest rank order compared to Work-items in later sprints or later release cycles. I will soon explain how this rank ordering is done.

The rank order information is used to decide the order in which the team will undertake work on Work-items in a sprint backlog, and also decide which incomplete Work-items to push out to the release or product backlog at the end of a sprint time-box.

Work-items in the next sprint collectively satisfy the well-known INVEST criteria; it is a meaningful English word, as well as an interesting acronym coined by Bill Wake. Its letters represent important characteristics of Work-items in the next sprint backlog. Stories in the next sprint backlog should be:

* **Independent of each other**: At the specification level stories are independent; they offer distinctly different functionality and don’t overlap. Moreover, at the implementation level these stories should also be as independent of each other as possible. However, sometimes certain implementation-level dependencies may be unavoidable.
* **Negotiable**: Stories in the next sprint are always subject to negotiations and clarifications among product owner (business proxy) and the members of agile development team.
* **Valuable**: Each story for the next sprint offers clear value or benefit to either external users or customers (outside the development team), or to the team itself, or to a stakeholder. For most products and projects, most stories offer value to external users or customers.
* **Estimable**: From the specification of story itself, an agile team should be able to estimate the effort needed to implement the story; this estimate is in relative size terms (story points), and optionally, it can also be in time units (such as ideal staff-hours or staff-days for the whole team). Thus, stories are estimated in story points, and also often in ideal time units.
* **Sized Appropriately**: A simpler interpretation of this criterion is that each story is Small enough to be completed and delivered in a single sprint. The letter “S” can be taken to mean Sized Appropriately; specifically, each story should take no more than N/4 staff-weeks of team effort for an N-week long sprint. Thus, for a 2-week sprint, each story should take no more than 2/4 staff-week = 0.5 staff-week = 20 staff-hours of effort. A story substantially larger than 20 staff-hours of total effort should be treated as an epic and be broken down into smaller stories. For a 4-week sprint, each story should take no more than 4/4 staff-week = 1 staff-week = 40 staff-hours of effort. If a sprint backlog has a mix of stories that are small, medium or large size stories (their average far exceeds N/4 staff-weeks), the average cycle time across all stories will increase dramatically reducing the team velocity.
* **Testable**: Each story specification is very clear to be able to develop all test cases from its acceptance criteria (which is part of the specification).

Stories may be broken down into implementation tasks, such as Analysis, Design, Code Development, Unit Testing, Test Case Development, On-line Help, etc. These tasks need to be SMART:

* + S: Specific
  + M: Measurable
  + A: Achievable
  + R: Relevant
  + T: Time-boxed (typically small enough to complete in a single day)

If a story needs to take no more than N/4 staff-week of team effort (ex. 20 staff-hours for 2-week sprints), all SMART tasks in a story should add up to no more than N/4 staff-week of team effort. If you have 5 tasks, each task on an average should take 4 hours of ideal time effort or less. Stories and its SMART tasks for the next sprint are worth investing in, as the return on that investment is high because they are scheduled to be worked on and delivered as working software in the next sprint itself.

# 3. RELEASE PLANNING AND WORK GRANURALITY

If the planning horizon is an upcoming release cycle (typically 8 to 26 weeks, or 2 to 6 months long – consisting of several sprints), Work-items are “medium-grain” as shown in the middle yellow colour region of Figure 1. Typically, many of these Work-items are epics; however, they should be still small enough to fit in a release cycle and can be completed over two or more sprints in a release cycle. These epics are typically called features or feature-epics. These feature-epics should still be specified with User Role, Action, Value and Acceptance Criteria formalism that is often used for specifying stories, but now you are capturing a larger functionality represented by a feature-epic. Feature-epics are divided into stories – small enough to fit in a sprint – before the sprint in which a story will be implemented.

Over the time horizon of an entire release cycle, investing in stories for an entire release cycle has poor returns, because it takes a lot of effort to ensure that the INVEST criteria is being satisfied correctly for a large number of stories covering an entire release cycle, and those stories are much more likely to change over the release cycle spanning several sprints; so this kind of investment may not yield expected results as stories will very likely change during an entire release cycle after they have been specified.

**Feature-epics** in a release cycle can and should be estimated in relative size terms, but without expending the effort needed to break down all feature-epics in a release cycle into individual stories. This epic-level estimation can be done by comparing relative sizes of epics.

It still makes sense to rank order feature-epics in a release cycle to decide which ones will be scheduled in Sprint 1, 2, 3, and so on. However, this assignment may change as each sprint is completed and more information and learning emerge.

# 4. PRODUCT PLANNING AND WORK-ITEM GRANURALITY

If the product planning horizon is over multiple release cycles (typically 6 to 24 months) going beyond the current release cycle, Work-items are “**coarse-grain**” as shown in the bottom grey colour region of Figure 1. These large epics or super epics require two or more release cycles to complete. These super epics may be described in plain English (bulleted text) or with screen mock-up or video or prototype or with any form of expression suitable to express the intent and value of super epics. These super epics are divided into feature-epics – small enough to fit in a single release cycle – before the release cycle in which that feature-epic will be implemented.

Over the time horizon of multiple release cycles, investing in stories has even poorer returns compared to investing in stories for a single release cycle. This kind of investment will not yield expected results as stories are very likely to change over much longer duration of multiple release cycles.

Large epics or super epics that need multiple release cycles to be implemented can and should be estimated in relative size terms, but without expending the effort needed to break down large epics into feature-epics, and breaking those, in turn, into stories.

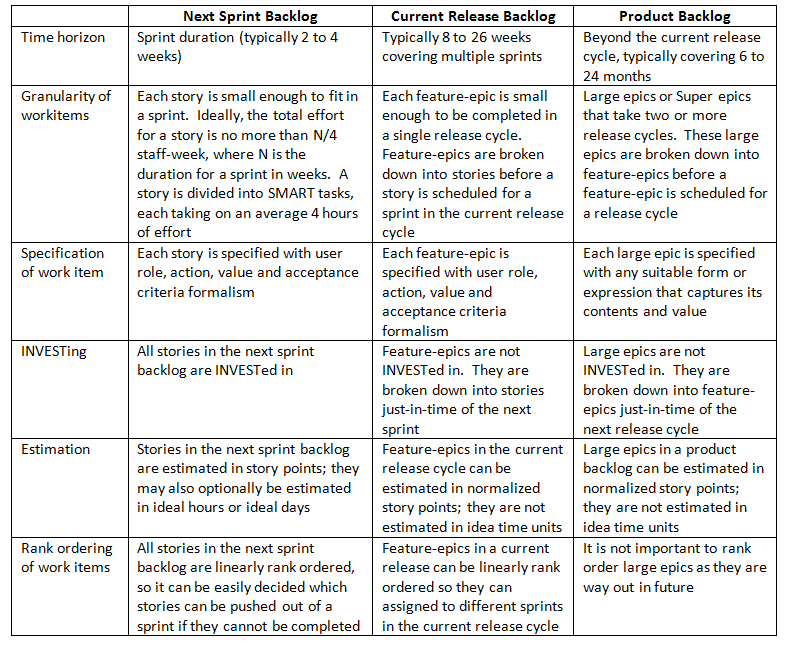
DIVE the product backlog carefully

There is rarely enough time or resources to do everything. Therefore, agile teams must prioritize (rank-order, to be more precise) which stories to focus on and which lowest rank-order stories could be pushed out of scope when close to the end of a sprint. For agile development projects, you should linearly rank-order the backlog, rather than do coarse-grain prioritization where stories and epics are lumped into a *small number of priority buckets, such as Low, Medium, High, Critical priorities. Linear rank ordering (i.e., 1, 2, 3, 4 ….n) avoids inflation of priority, keeps everyone honest, and forces decisions on what is really important. It discourages the “kid-in-a-candy-shop” behaviour when the business side clamours that everything is of high-priority or of equal importance.*

*Note that epics and stories are conceptually different, and should not be mixed or aggregated while developing a rank order. An epic rank order is separate from a story rank order.*

*The responsibility of agile rank ordering is shared among all members of a team; however, the rank ordering effort is led by the product owner. Similar to DEEP, INVEST and SMART, DIVE is a meaningful English word, and also an acronym. Product backlog items should be linearly ordered based on the DIVE criteria, which requires careful consideration of all four factors captured in the DIVE acronym:*

* *Dependencies: Even after minimizing the dependencies among stories or epics (which is always a good thing to do), there may still be few unavoidable dependencies and they will have an impact on rank ordering. If Work-item A depends on B, B needs to be rank-ordered higher than A.*
* *Insure against Risks: Business as well as technical risks*
* *Business Value*
* *Estimated Effort*



# 5. PRODUCT BACKLOG: GOALS GRANURALITY

|  |  |
| --- | --- |
| Goal-ID-1 | Build Farmers Profile |
| Purpose | Build Farmers Profile so that they can use the functionality of web application |
| Target Audience | Farmers |
| Status | Completed |
| Task Description | 1. Filter Data Sources |
|  | 2. Collect Farmers Data |
|  | 3. Ascertain Data Correctness |
|  | 4. Identify appropriate DBMS |
|  | 5. Insert Farmers Data into database |
|  | 6. Create Cloud Backup of Database |
|  | 7. Create Farmer profile |
|  | 8. Display weather details to Farmer profile |
|  | 9. Display crop-cycle to the Farmer profile |
|  | 10. Identify the attributes of Farmers to Store in the database. |
|  | 11. Sort farmers according to crop type |
|  | 12. Farmers has to add his location and land area. |
|  | 13. Ads posting is suggested |
|  | 14. Posting ads can be scheduled |
|  | 15. Ad is posted in my crops |
|  | 16. Farmers can see his ads in his profile. |

|  |  |
| --- | --- |
| Goal-ID-2 | Build Market agent (Customer) Profile |
| Purpose | The purpose is to develop Customer Profile to enable the system to use that information later. |
| Target Audience | Customers |
| Status | Completed |
| Task Description | 1. Create Customer profile |
|  | 2.Identify the attributes of Customer to store in database |
|  | 3.Create Cloud Backup of database |
|  | 4. Create UI for Customer profile |
|  | 5. Create Registration process for new Customer |
|  | 6.Display Advertisement of farmer’s crop production Category wise to Customer |
|  | 7. Display Advertisement of nearby farmers only |
|  | 8. Send request to farmer after Customer buy the production |
|  | 9. Receive notification of order accepted after farmer accepted the order. |
|  | 10. Create customer edit profile page |
|  | 11. Display updated profile page. |
|  | 12. Display required farmers details |
|  | 13. Display market details |
|  | 14. Contact of farmers |
|  | 15. Nearby dealers are displayed |
|  | 16. Overall market data is displayed. |

|  |  |
| --- | --- |
| Goal-ID-3 | Perform Add Crop production for selling |
| Purpose | The purpose to add crop production is to get good price foe the farmers crop |
| Target Audience | Farmers and Customers |
| Status | Completed |
| Task Description | 1. Display Crop to Farmer in the form of images for selection |
|  | 2.Farmers will select crop and add quantity and price and post the advertisement. |
|  | 3.Farmers can see their posted advertisement. |
|  | 4.Farmers can update their advertisement. |
|  | 5. Farmers can delete their advertisement. |
|  | 6.Farmers get request for their advertisement from Customers. |
|  | 7. Farmers have to either accept or decline the order request. |
|  | 8. Send notification to Customer after accepting or declining order request. |
|  | 9. Receive notification of order accepted after farmer accepted the order. |
|  | 10. Get contact information of Customer who requested to their advertisement. |

|  |  |
| --- | --- |
| Goal-ID-4 | Customer buy crop production |
| Purpose | The purpose to buy crop production from this portal to promote e-marketing of crop production |
| Target Audience | Farmers and Customers |
| Status | Completed |
| Task Description | 1. Display Crop production advertisement Category-wise. |
|  | 2.Show advertisement of nearby Farmers only. |
|  | 3.Customer send Request to farmers for buying crop production. |
|  | 4.Recieve notification to customer after farmer accept or decline the order. |
|  | 5. Show Customer all their orders in one place only. |
|  | 6.Display Farmer details to Customer after buying the crop production. |
|  | 7.Display total bill of the order to Customer. |
|  | 8. Send request to farmer after Customer buy the production |
|  | 9. Receive notification of order accepted after farmer accepted the order. |
|  | 10. Customers can see the crop details of ads posted |
|  | 11. Photos and land information are displayed |
|  | 12. He can contact farmers. |
|  | 13.Orders can be edited |
|  | 14. Ads can be edited and updated with latest information |
|  | 15. The latest orders are taken and order is placed. |
|  | 16. Order confirmation and verification is done. |

|  |  |
| --- | --- |
| Goal-ID-6 | Display weather |
| Purpose | The purpose to display weather information crop. |
| Target Audience | Farmers |
| Status | Completed |
| Task Description | 1. Display weekly weather. |
|  | 2. To display weather according to time. |
|  | 3. Farmers can see if rains are coming. |
|  | 4. Weather API’s are used for weather information. |
|  | 5. API are converted into a GUI. |
|  | 6. Weather is displayed in the given GUI. |
|  | 7. GUI has features like time wise, week wise. |
|  | 8. Any type of seasonal change. |
|  | 9. If it rains it will show raining animation. |
|  | 10. Notifications are send for weather change. |
|  | 11. Help and guidance will be given according to weather change. |
|  | 12. farmers can contact weather experts. |
|  | 13. Weather experts help farmer. |
|  | 14. Weather for another village can be also seen by the farmer |
|  | 15. The latest weather is taken and displayed. |
|  | 16. The main objective of displaying weather is to alert farmer from upcoming weather conditions. |

|  |  |
| --- | --- |
| Goal-ID-5 | Display crop cycle |
| Purpose | The purpose to display crop cycle. |
| Target Audience | Farmers |
| Status | Completed |
| Task Description | 1. Display selected crop by the farmer for Seeding |
|  | 2. Display total time for harvesting of selected crop |
|  | 3. Take Crop information from the Database |
|  | 4. Display Information of the Crop Acquired |
|  | 5. Collect the Precautions methods stored in Database |
|  | 6. Display The precautions methods for the crop before harvesting. |
|  | 7. Display Time to Time precautions methods for the crop after Seeding. |
|  | 8. Collect Input from the Farmer about Crop after seeding after appropriate time. |
|  | 9. Collect the input From the Farmer About crop Harvested. |
|  | 10. Notify Farmer about the crop conditions using his feedback. |
|  | 11. Help and guide can be provided to the farmer if required according to the data collected |
|  | 12. Create GUI for the Crop Cycle. |
|  | 13. After Every time interval GUI will alert farmer to take necessary steps. |
|  | 14.At the Time of harvesting app will show alert notifications |
|  | 15. After harvesting new crop can be added for the seeding. |
|  | 16. Feedback will be taken after every crop is been harvested. |

T.Y. B. Tech.

CS 303: Software Engineering Laboratory

Assignment No: 7

**MOOC APP**

# User Story Cards

**16-04-2019**

**Version 1.0**

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front End** |
| 62 | **161259** | **Aniket Thaware** | **Backend** |
| 73 | **161854** | **Rushabh pahade** | **Leader** |

## **Approved By: Mahesh R. Dube**

## Academic Year: 2018-19 Semester: II

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction | **66** |
| 2 | Epics and User Stories | **66** |
| 3 | User Stories-Goal 1 | **68** |
| 4 | User Stories-Goal 2 | **70** |
| 5 | User Stories-Goal 3 | **72** |
| 6 | User Stories-Goal 4 | **74** |
| 7 | User Stories-Goal 5 | **76** |
| 8 | User Stories-Goal 6 | **79** |
| 9 | User Stories-Goal 7 | **81** |

## 1. INTRODUCTION

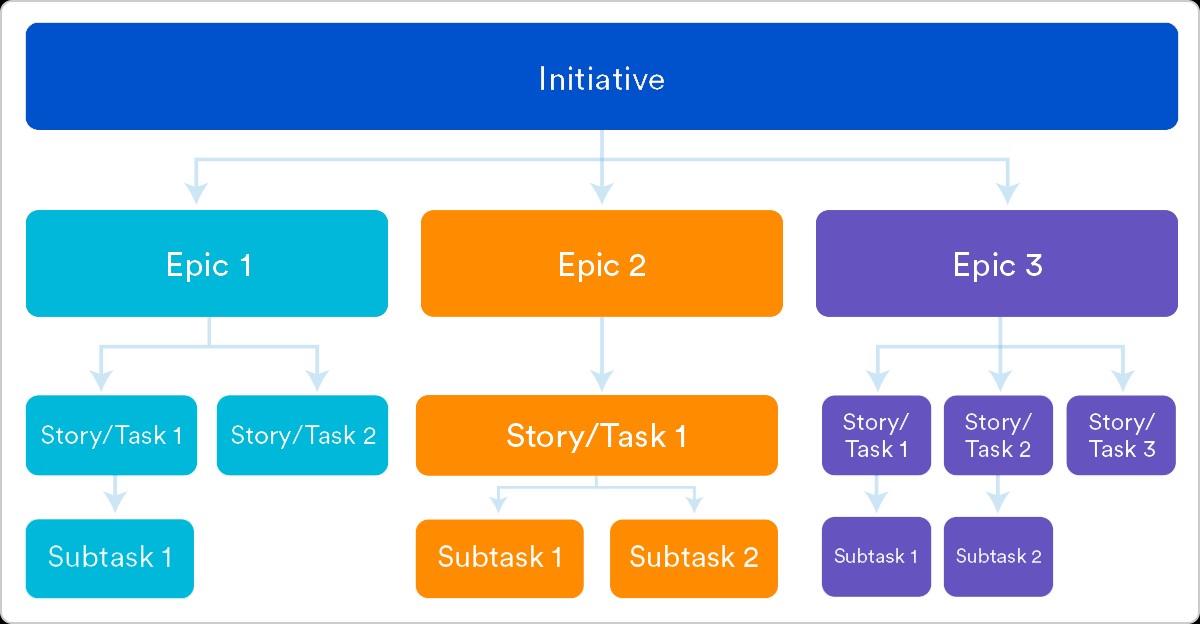
What does defining customer problems look like in an agile world? The agile manifesto reminds us that we don’t always have to do it the “traditional” way. As product managers, we should be doing whatever is required to tell the story of the customer. Try different things: experiment, explore, then do what works best for you and your team in the context that you might be working in.

* If it means you can have several discussions and sketch something on a bit of paper – then do it.
* What if you could get everyone (including the customer) in a room and do a user story mapping exercise? If that communicates the problems well, then you don’t need to go much further.
* Or what if you can visit the customer and watch them use your product in context? Could you get your engineers and designers to sit next to the customer to listen to and observe their problems?
* Instrumenting your product with analytics hooks give you aggregate, concrete data about how customers as a whole are using your product.
* Another option would be to grab the product triad (a product manager, engineer and a designer) for a quick stand-up to sketch, discuss and make some quick decisions on the spot.
* Need to explore some more? Try running a workshop where you gather key stakeholders and do lots and lots of white-boarding or even paper prototyping to dive deep into understanding the problems you are trying to solve and how you could solve those problems.

|  |  |  |  |
| --- | --- | --- | --- |
| **Epic**  Large body of work, contains stories | **Story**  Smallest unit of work, also known as a task | **Version**  The release of software to the customer | **Sprint**  Iteration where team does the work |

## 2. EPICS AND USER STORIES

Epics are larger bodies of work that stories roll up into. An epic can span across multiple sprints and versions. Versions are different from epics, because they are a point in time where software is released to the customer. A version might contain multiple epics. Epics help teams create hierarchy and structure. Stories help teams keep track of specific details for the task at hand and can be broken down into sub-tasks.



 An **epic** is a large body of work that can be broken down into a number of smaller stories. For example, performance-related work in a release. An epic can span more than one project, if multiple projects are included in the board to which the epic belongs.

 Unlike sprints, epics often change in scope over time as a natural aspect of agile development. Epics are almost always delivered over a set of sprints. As a team learns more about an epic through development and customer feedback, user stories will be added and removed to optimize the team's release time.

 **Burndown** **charts** can also be used to visualize epics, which keep teams motivated and the executive stakeholders informed. A good epic burndown chart shows the agile nature of development. It's clear how the team is progressing as well as where the product owner added and removed user stories. Having these data points clearly visible keeps everyone on the same page and facilitates open conversation about the evolution of the product and completion forecasts. Not to mention that transparency builds trust!

 A story or **user story** is the smallest unit of work in an agile framework. It is a software system requirement that is expressed in a few short sentences, ideally using non-technical language.

 The goal of a user story is to deliver a particular value back to the customer. Note that "customers" don't have to be external end users in the traditional sense, they can also be internal customers or colleagues within your organization who depend on your team.

 **User stories** are a few sentences in simple language that outline the desired outcome. They don't go into detailed requirements.

 **Versions** are the actual releases of software out to customers. Remember, at the end of each sprint the team should be able to ship the software to customers. Versions are the curated changes the product owner actually ships.

 **Versions** are often developed over a set of sprints, much like epics. Savvy product owners may choose to deliver an epic over several versions. An epic does not have to be fully contained within a version. By delivering an epic over several versions, the product owner can learn how the market is responding to that epic and make calculated decisions about its future direction rather than doing one giant release.

 A **sprint** is a short period in which the development team implements and delivers a discrete and potentially shippable application increment, e.g. a working milestone version. If you haven't run sprints before, we recommend using a fixed two-week duration for each sprint. It's long enough to get something accomplished, but not so long that the team isn't getting regular feedback.

 In **scrum**, teams commit to complete a set of user stories during a fixed time period. Generally speaking, sprints are one, two, or four weeks long. It's up to the team to determine the length of a sprint. Once a sprint cadence is determined, the team perpetually operates on that cadence. Fixed length sprints reinforce estimation skills and enable the ability to predict the future **velocity** for the team once they have the data from several completed sprints.

Once a team commits to a set of user stories for the sprint, and the sprint is started, the scrum master is in charge of fending off changes to the user stories. This keeps the team focused and combats "s**cope creep**" (adding work to the sprint after the sprint starts). Adding work mid-sprint compromises the team's ability to forecast and estimate accurately.

At the end of each sprint, the team is required to deliver a working piece of software. In scrum, that's called a **potentially shippable increment** (PSI). The product owner ultimately decides when the PSI gets released to customers, but the work should be complete enough to be suitable for release at the end of the sprint.

In agile development, **work in progress** (WIP) limits set the maximum amount of work that can exist in each status of a workflow. Limiting the amount of work in progress makes it easier to identify inefficiency in a team's workflow. Bottlenecks in a team's delivery pipeline are clearly visible before a situation becomes dire

### **USER STORIES: GOAL-1: BUILD FARMERS PROFILE**

|  |  |  |
| --- | --- | --- |
| **Objective-1** | **Create Farmers Profile** | |
| **Purpose** | The purpose is to develop Farmer Profile to enable the system to use that information later. | |
| **Target Audience** | Farmers | |
| **Status** | Completed | |
| **Role:** | **As a**developer | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Request creation of Farmer Profile | use it to do further activities. |
|  | 2. Collect Farmer’s data | Build Farmer profiles. |
|  | 3. Create a preliminary database | can store the Farmer’s data. |
|  | 4. Create a Farmer Data Spyder | get updated Farmer data. |
|  | 5. Formulate database structure | start creating profiles. |
|  | 6. Populate Farmer Database | meet the preliminary objective. |
|  | 7. Generate a backup | retrieve data in case of loss of files. |
|  | 8. Share backup with Project Team | expect team to perform assigned tasks. |
|  | 9. Assign database privileges | monitor the changes made to the database. |
|  | 10. Launch Farmer Profile Page | fulfil project deliverables. |

|  |  |  |
| --- | --- | --- |
| Process-1 | Collect Farmers Data. | |
| Purpose | Before using the functionalities of the web application farmers data must be available in database. | |
| Target Audience | Farmers | |
| Status | Completed | |
| Role: | **As a**developer | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| Task Description | 1.Create UI to collect Farmer details | Receive appropriate information from the Farmer |
|  | 2.Acquire information criterion from the company | Obtain only that information which is important to the company |
|  | 3.Provide UI for details to every Farmer | Collect the information of Farmer |
|  | 4. Set up a mandatory field set in the Farmer UI | Collect important details of Farmer |
|  | 5. Add proper data types for the asked information | Store the information in the right format |
|  | 6. Provide service of uploading documents in the Farmer UI | Preserve relevant documents |
|  | 7. Provide easy way to enter dates in the information section | Maintain a proper format for dates |
|  | 8. Transfer collected data for storage | Preserve this Farmer information for future references |
|  | 9. Create log file to record every change made in the system | Store the changes made in the system |
|  | 10. Present the Farmer data to the company | Make it easier to spot |

|  |  |  |
| --- | --- | --- |
| Process-2 | Validate Collected Data. | |
| Purpose | The information provided by the Farmers is validated by the system so that appropriate and correct information is stored in the database of Farmer information | |
| Target Audience | Internal stakeholders | |
| Status | Completed | |
| Role: | **As a**developer | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| Task Description | 1.Confirm if data in CV is same as data provided in the information section | Provide validated information to the project team |
|  | 2. Confirm that data is stored in the correct column | Provide reliable information to the company |
|  | 3. Check if a profile has been repeated | Avoid space wastage |
|  | 4. Check if the mandatory fields have been filled | Make sure important information is collected |
|  | 5. Validate the entered information by the Farmer | Provide reliable information to the project team |
|  | 6. Delete irrelevant information of unselected candidates after the selection | Avoid space wastage |
|  | 7. Validate trueness of uploaded documents | Remove the fake Farmer profiles |
|  | 8. Inform the glitches to the respective Farmer | Ensure that the required changes are made |
|  | 9. Maintain the record of the changes made to the database | Keep track of every action done by the system |
|  | 10.Provide the record of changes to the team | Inform the company of every change in the records |

|  |  |  |
| --- | --- | --- |
| **Objective-2** | **Analyse Farmer’s Data** | |
| **Purpose** | To decide the Farmer analysis process of the system and the methodology to follow. | |
| **Target Audience** | Internal Stakeholders | |
| **Status** | Completed | |
| **Role:** | **As a**developer | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Organise database attributes | easily analyse the data. |
|  | 2. Design patterns for attributes | study the data. |
|  | 3. Conceptualise output parameters | process of analysing is directed. |
|  | 4. Prioritise the important parameters | produce accurate results. |
|  | 5. Draw a map for the shortlisted attributes | formalize the observations. |
|  | 6. Organise the parameters | simplify analysis process. |
|  | 7. Record formulated observations | discuss with the analysis team. |
|  | 8. Correspond with Analysis team | refine the observations |
|  | 9. Consolidate outline of analysis process | systemize procedure. |
|  | 10. Construct final analysis methodology | begin development process. |

|  |  |  |
| --- | --- | --- |
| **Process-1** | Identify cause of Farmers Problem | |
| **Purpose** | To re-iterate data from Database  To extract irrelevant information. | |
| **Target Audience** | Stakeholders | |
| **Status** | Completed | |
| **Role:** | **As a**developer | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Traverse Farmer Information | Acquire relatable keywords |
|  | 2. Observe for any causes defined | Identify Farmer Query causes. |
|  | 3. Record Formulated Observations. | Discuss with the analysis team. |
|  | 4. Correspond with the analysis team | Refine the Observation. |
|  | 5. Deploy it to Analyst to check | Identify Query cause |
|  | 6. Assess result arbitrated by Analyst | Produce accurate result |
|  | 7. Consolidate outline of Analysis methodology | Estimate causes of the Problem |
|  | 8. Construct final Analysis methodology | Work upon the methodology |

|  |  |  |
| --- | --- | --- |
| **Process-2** | Ascertain Correctness of Data | |
| **Purpose** | To verify stored Information is accurate | |
| **Target Audience** | Stakeholders | |
| **Status** | On-going | |
| **Role:** | **As a** *developer* | |
|  | **I want to** *ascertain correctness of Data* | **I want to** *ascertain correctness of Data* |
| **Task Description** | 1. Determine domain of Query described | 1. Determine domain of Query described |
|  | 2. Analyse terms used by User | 2. Analyse terms used by User |
|  | 3. Interpret meaning of the User terms | 3. Interpret meaning of the User terms |
|  | 4. Reconstruct phrases | 4. Reconstruct phrases |
|  | 5. Conceive exact Query of the User | 5. Conceive exact Query of the User |
|  | 6. Map conceived terms to stored Data | 6. Map conceived terms to stored Data |
|  | 7. Deploy Data to find Solution | 7. Deploy Data to find Solution |
|  | Ascertain Correctness of Data | Ascertain Correctness of Data |

#### USER STORIES: GOAL-2: BUILD Customer Profile

|  |  |  |
| --- | --- | --- |
| **Objective-1** | Create Customer Profile | |
| **Purpose** | Make separate database for Customer. | |
| **Target Audience** | Customers | |
| **Status** | Completed | |
| **Role:** | **As a customer** | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Request creation of Customer Profile | use it to do further activities. |
|  | 2. Collect Customer data | Build Customer profiles. |
|  | 3. Create a preliminary database | can store the Customer data. |
|  | 4. Create a Customer Data Spyder | get updated Customer data. |
|  | 5. Formulate database structure | start creating profiles. |
|  | 6. Populate Customer Database | meet the preliminary objective. |
|  | 7. Generate a backup | retrieve data in case of loss of files. |
|  | 8. Share backup with Project Team | expect team to perform assigned tasks. |
|  | 9. Assign database privileges | monitor the changes made to the database. |
|  | 10. Launch Customer Profile Page | fulfil project deliverables. |

|  |  |  |
| --- | --- | --- |
| **Process-1** | Collect Customer Data. | |
| **Purpose** | Before using the functionalities of the web application Customer data must be available in database. | |
| **Target Audience** | Farmers | |
| **Status** | Completed | |
| **Role:** | **As a**developer | |
|  | **I want to** *<perform some task>* | **I want to** *<perform some task>* |
| **Task Description** | 1.Create UI to collect Customer details | 1.Create UI to collect Customer details |
|  | 2.Acquire information criterion from the company | 2.Acquire information criterion from the company |
|  | 3.Provide UI for details to every Customer | 3.Provide UI for details to every Farmer |
|  | 4. Set up a mandatory field set in the Farmer UI | 4. Set up a mandatory field set in the Farmer UI |
|  | 5. Add proper data types for the asked information | 5. Add proper data types for the asked information |
|  | 6. Provide service of uploading documents in the Customer UI | 6. Provide service of uploading documents in the Farmer UI |
|  | 7. Provide easy way to enter dates in the information section | 7. Provide easy way to enter dates in the information section |
|  | 8. Transfer collected data for storage | 8. Transfer collected data for storage |
|  | 9. Create log file to record every change made in the system | 9. Create log file to record every change made in the system |
|  | 10. Present the Customer data to the company | 10. Present the Customer data to the company |

|  |  |  |
| --- | --- | --- |
| **Process-2** | Validate Collected Data. | |
| **Purpose** | The information provided by the Customer is validated by the system so that appropriate and correct information is stored in the database of Farmer information | |
| **Target Audience** | Internal stakeholders | |
| **Status** | Completed | |
| **Role:** | **As a**developer | |
|  | **I want to** *<perform some task>* | **I want to** *<perform some task>* |
| **Task Description** | 1.Confirm if data in CV is same as data provided in the information section | 1.Confirm if data in CV is same as data provided in the information section |
|  | 2. Confirm that data is stored in the correct column | 2. Confirm that data is stored in the correct column |
|  | 3. Check if a profile has been repeated | 3. Check if a profile has been repeated |
|  | 4. Check if the mandatory fields have been filled | 4. Check if the mandatory fields have been filled |
|  | 5. Validate the entered information by the Farmer | 5. Validate the entered information by the Farmer |
|  | 6. Delete irrelevant information of unselected candidates after the selection | 6. Delete irrelevant information of unselected candidates after the selection |
|  | Validate Collected Data. | Validate Collected Data. |
|  | The information provided by the Farmers is validated by the system so that appropriate and correct information is stored in the database of Farmer information | The information provided by the Farmers is validated by the system so that appropriate and correct information is stored in the database of Farmer information |
|  | Internal stakeholders | Internal stakeholders |
|  | Completed | Completed |

|  |  |  |
| --- | --- | --- |
| **Objective-2** | Analyse Customer Data |  |
| **Purpose** | To decide the Customer analysis process of the system and the methodology to follow. |  |
| **Target Audience** | Customers |  |
| **Status** | Completed |  |
| **Role:** | **As a**Developer |  |
|  | **I want to** *<perform some task>* | **I want to** *<perform some task>* |
| **Task Description** | 1. Organise database attributes | easily analyse the data. |
|  | 2. Design patterns for attributes | study the data. |
|  | 3. Conceptualise output parameters | process of analysing is directed. |
|  | 4. Prioritise the important parameters | produce accurate results. |
|  | 5. Draw a map for the shortlisted attributes | formalize the observations. |
|  | 6. Organise the parameters | simplify analysis process. |
|  | 7. Record formulated observations | discuss with the analysis team. |
|  | 8. Correspond with Analysis team | refine the observations |
|  | 9. Consolidate outline of analysis process | systemize procedure. |
|  | 10. Construct final analysis methodology | begin development process. |

|  |  |  |
| --- | --- | --- |
| **Process-1** | Identify cause of Customer Problem | |
| **Purpose** | To re-iterate data from Database  To extract irrelevant information. | |
| **Target Audience** | Stakeholders | |
| **Status** | Completed | |
| **Role:** | **As a**developer | |
|  | **I want to** *<perform some task>* | **I want to** *<perform some task>* |
| **Task Description** | 1. Traverse Customer Information | 1. Traverse Customer Information |
|  | 2. Observe for any causes defined | 2. Observe for any causes defined |
|  | 3. Record Formulated Observations. | 3. Record Formulated Observations. |
|  | 4. Correspond with the analysis team | 4. Correspond with the analysis team |
|  | 5. Deploy it to Analyst to check | 5. Deploy it to Analyst to check |
|  | 6. Assess result arbitrated by Analyst | 6. Assess result arbitrated by Analyst |
|  | 7. Consolidate outline of Analysis methodology | 7. Consolidate outline of Analysis methodology |
|  | 8. Construct final Analysis methodology | 8. Construct final Analysis methodology |
|  | Identify cause of Farmers Problem | Identify cause of Farmers Problem |
|  | To re-iterate data from Database  To extract irrelevant information. | To re-iterate data from Database  To extract irrelevant information. |

|  |  |  |
| --- | --- | --- |
| **Process-2** | Ascertain Correctness of Data | |
| **Purpose** | To verify stored Information is accurate | |
| **Target Audience** | Stakeholders | |
| **Status** | On-going | |
| **Role:** | **As a** *developer* | |
|  | **I want to** *ascertain correctness of Data* | **I want to** *ascertain correctness of Data* |
| **Task Description** | 1. Determine domain of Query described | 1. Determine domain of Query described |
|  | 2. Analyse terms used by User | 2. Analyse terms used by User |
|  | 3. Interpret meaning of the User terms | 3. Interpret meaning of the User terms |
|  | 4. Reconstruct phrases | 4. Reconstruct phrases |
|  | 5. Conceive exact Query of the User | 5. Conceive exact Query of the User |
|  | 6. Map conceived terms to stored Data | 6. Map conceived terms to stored Data |
|  | 7. Deploy Data to find Solution | 7. Deploy Data to find Solution |
|  | Ascertain Correctness of Data | Ascertain Correctness of Data |
|  | Ascertain Correctness of Data | Ascertain Correctness of Data |
|  | To verify stored Information is accurate | To verify stored Information is accurate |

#### USER STORIES: GOAL-3: PERFORM ADD CROP PRODUCTION FOR SELLING

|  |  |  |
| --- | --- | --- |
| **Objective-1** | To make farmers profile | |
| **Purpose** | To extract customers’s data from database and performed operation on received information. | |
| **Target Audience** | Customers and Farmers | |
| **Status** | Completed | |
| **Role:** | **As a**Developer | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Request creation of customers’s profile | use it to do further activities. |
|  | 2. Collect customers’s data | build customers profile. |
|  | 3. Create a preliminary database | can store customers’s data. |
|  | 4. Create a customers’s Data Spyder | can examine data. |
|  | 5. Formulate database structure | start creating profile. |
|  | 6. Populate Customers’s database | meet the preliminary objective |
|  | 7. Generate a backup | retrieve data in case of loss of files. |
|  | 8. Share backup with Project Team | expect team to perform assigned tasks. |
|  | 9. Assign database privileges | monitor the changes made to the database. |
|  | 10. Launch Farmer Profile Page | fulfil project deliverables. |

|  |  |  |
| --- | --- | --- |
| **Objective-2** | **Analyse farmers Data** | |
| **Purpose** | To decide the customers analysis process of the system and the methodology to follow. | |
| **Target Audience** | Internal Stakeholders | |
| **Status** | Completed | |
| **Role:** | **As a Developer** | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Organise database attributes | easily analyse the data. |
|  | 2. Design patterns for attributes | study the data. |
|  | 3. Conceptualise output parameters | process of analysing is directed. |
|  | 4. Prioritise the important parameters | produce accurate results. |
|  | 5. Draw a map for the shortlisted attributes | formalize the observations. |
|  | 6. Organise the parameters | simplify analysis process. |
|  | 7. Record formulated observations | discuss with the analysis team. |
|  | 8. Correspond with Analysis team | refine the observations |
|  | 9. Consolidate outline of analysis process | systemize procedure. |
|  | 10. Construct final analysis methodology | begin development process. |

USER STORIES: GOAL-4: CUSTOMER BUY CROP PRODUCTION

|  |  |  |
| --- | --- | --- |
| **Objective-1** | **Perform Customer related activities.** |  |
| **Purpose** | Customers will receive the crop details. |  |
| **Target Audience** | Customers |  |
| **Status** | Completed |  |
| **Role:** | **As a customers** |  |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Register all customers to this app | can take advantage of e-marketing platform. |
|  | 2. Retrieve Crop advertisement. | Can view uploaded advertisement online |
|  | 3. Retrieve notifications. | Can see uploaded document online. |
|  | 4. Submit feedback to Farmer | Farmer can see who give order . |
|  | 5. Provide prices to the customer of advertisement. | Customers get to know their marks of submitted assignments. |
|  | 6. Get notifications. | Customers know about important notice. |
|  | 7. Get category wise advertisement. | Get all advertisement specific online. |
|  | 8. Buy from any Category | Can see orders in my order. |
|  | 9. Ask doubts to Farmer for any doubt | In the form of comments. |
|  | 10. Receive Farmer reply. | So that doubts can be clear. |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Objective-2** | **Classify Customers** | |
| **Purpose** | To classify customers | |
| **Target Audience** | Customers | |
| **Status** | Completed | |
| **Role:** | **As a** Developer | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1.Get customers’s attributes and data | Identify customers |
|  | 2.Examine data distribution per attribute | View relationships between the data |
|  | 3.Fix inconsistencies in selected attributes | Ensure correct data is used in future processes. |
|  | 4.Prototype groups according to attributes | Select viable grouping attributes |
|  | 5.Receive subject wise content | Enable grouped data usage in further processes |
|  | 6.Access study material groups | View grouped data |
|  | 7.Form a basis/gist for the group | The usage of the group is enabled |
|  | 8.Choose encoding method for the basis | Encode data appropriately |
|  | 9.Establish results of encoding analysis | Forward it to developers |
|  | 10.Integrate results | They can be used in modelling. |

USER STORIES: GOAL-5: DISPLAY WEATHER

|  |  |  |
| --- | --- | --- |
| **Objective-1** | Display weather information | |
| **Purpose** | To make weather information effective. | |
| **Target Audience** | Famer | |
| **Status** | On-going | |
| **Role:** | **As a** *developer* | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Upload location wise weather | Customers can have playlist of video tutorials. |
|  | 2. Implement AI module | To make better search. |
|  | 3. Provide input to search | Parameter for search |
|  | 4. Implement level wise search | By matching second third alphabet likewise. |
|  | 5. Implement level wise search | By matching second third words likewise. |
|  | 6. Record weather details | Store and compare them. |
|  | 7. Optimize search time | To do fast search |
|  | 8. Record most frequent search queries | For better and fast search |

|  |  |  |
| --- | --- | --- |
| **Objective-2** | Categorize weather information | |
| **Purpose** | To display weather animation to farmer | |
| **Target Audience** | farmer | |
| **Status** | On-going | |
| **Role:** | **As a** *Developer* | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1.Get search attributes and data | Identify video |
|  | 2.Examine data distribution per attribute | View relationships between the data |
|  | 3.Fix inconsistencies in selected attributes | Ensure correct data is used in future processes. |
|  | 4.Prototype groups according to attributes | Select viable grouping attributes |
|  | 5.Receive subject wise content | Enable grouped data usage in further processes |
|  | 6.Access location wise weather | View grouped data |
|  | 7.Form a basis/gist for the group | The usage of the group is enabled |
|  | 8.Choose encoding method for the basis | Encode data appropriately |
|  | 9.Establish results of encoding analysis | Forward it to developers |
|  | 10.Integrate results | They can be used in modelling. |

USER STORIES: GOAL-6: DISPLAY CROP CYCLE.

|  |  |  |
| --- | --- | --- |
| **Objective-1** | To show progress of seeded crop. | |
| **Purpose** | To display crop cycle | |
| **Target Audience** | Famers and customers | |
| **Status** | Completed | |
| **Role:** | **As a** *farmer* | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1.Prepare each step of growing crop. | Append in database if its rights |
|  | 2.Get topic’s data from respected source | Append in database for more info |
|  | 3. Implement time clock | To allot time slot to each question |
|  | 4.Create database for all crops. | To check crop selected by farmer right or wrong. |
|  | 5.Create database for uploaded crop by farmer | Will deliver to customers after uploading crop seeding date |
|  | 6. Retrieve crop submitted by customers | For display purpose |
|  | 7. Archive all crops which are not selected | Clear Clutter |
|  | 8. Send feedback to system | System will know their feedback |

|  |  |  |
| --- | --- | --- |
| **Objective-2** | To display crop cycle according to crop type. | |
| **-Purpose** | To display crop cycle | |
| **Target Audience** | Farmers. | |
| **Status** | Completed | |
| **Role:** | **As a**farmers | |
|  | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| **Task Description** | 1. Receive crop cycle from registered crop by Farmer | Can see crop cycle |
|  | 2. Farmers login through its credentials | Will help Farmer to identify the Crops |
|  | 3. Farmers can get indication for each step of growing | To know each step of growing |
|  | 4. Farmers can get information for each step separately. | To know each step separately. |
|  | 5. Time clock should be visible to Farmers | customers will know how much time is left. |
|  | 6. Farmers can submit crop feedback | For evaluation purpose |

**T.Y. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 9

**ERP-Farmer**

**System Construction**

***16.04.19***

***Version 1.0***

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front End** |
| 62 | **161259** | **Aniket Thaware** | **Backend** |
| 73 | **161854** | **Rushabh pahade** | **Leader** |

**Approved By:**

**Academic Year: 2018-19 Semester: II**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction | **65** |
| 2 | Architecture Objectives | **68** |
| 3 | System Design Specification | **69** |

# 1. INTRODUCTION

# The software engineering community realized that software architecture is not only about structures (components and interfaces), but also about system behavior (interaction between components, protocols). Furthermore, this community introduced an architectural design phase in the system life cycle, in which requirements should be satisfied and which should serve as a basis for detailed design activities. Researchers and engineers in software engineering have adopted the term 'architecture' as well. Nevertheless, there is no consensus about the subject; no universally-accepted definition of the term 'architecture' is agreed upon.

# Perry and Wolf (1992) consider a software architecture as a set of architectural elements that have a particular form. Similar to Zachman and Van Waes, they distinguish three different classes of architectural elements: processing, data, and connecting elements. Perry and Wolf consider an architecture as a necessary framework in which requirements are satisfied and which serves as a basis for the design.

# Garlan et al. (1995) stated that a system's architectural design is concerned with describing its decomposition into computational elements and their interactions. Design tasks at this level include organizing the system as a composition of components; developing global control structures; selecting protocols for communication, synchronization, and data access; assigning functionality to design elements; physically distributing the components; scaling the system and estimating performance; defining the expected evolutionary paths; and selecting among design alternatives.

# Soni et al. (1995) stated that software architecture is concerned with capturing the structures of a system and the relationships among the elements both within and between structures. Software architectures describe how a system is decomposed into components, how these components are interconnected, and how they communicate and interact with each other. Based on a survey on the role of architecture in the design and development of large systems within Siemens, Soni et al. notice that different structures are used at different stages of the development process. Each structure describes the system from a different perspective.

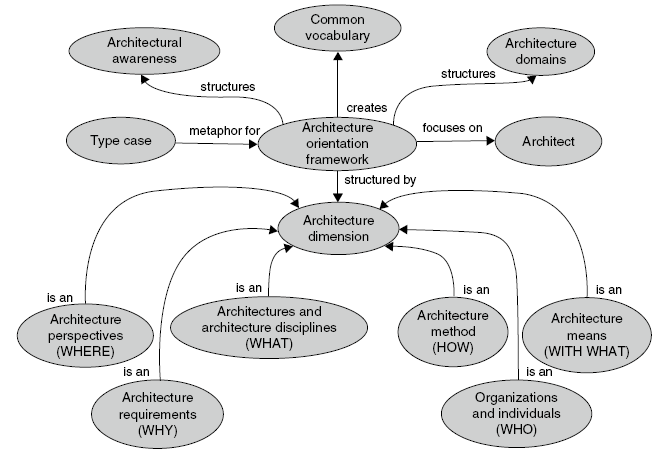
# Soni et al. argue that the four different architectures they distinguished are needed because of the growing complexity of software throughout history (see Figure 1.3). Initially, only the code architecture was required. The module and execution architecture became necessary when systems became larger and distributed. Now, software engineers would like to use communicating objects and assemblies of reused components. Therefore, a high-level structure is described in the form of a conceptual architecture. On the other hand, Zachman and especially Van Waes reason that their various architectures are wanted as representation for each of the involved actors.

# Garlan and Perry (1995) found that the term 'architecture' is used in a number of ways in software engineering. Among the various uses are a) the architecture of a particular system, as in 'the architecture of this system consists of the following three components,' b) an architectural style, as in 'this system adopts a client-server architecture,' and c) the general study of architecture, as in 'the papers in that issue are about architecture.'

# A discussion group at Carnegie Mellon University's Software Engineering Institute developed a typical definition: the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time. They represent a spectrum in the software architecture community about the emphasis that should be placed on architecture - its constituent parts, the whole entity, the way it behaves once built, or the process of building it. Taken together, they reflect the various aspects of software architecture.

# Software architecture is concerned with the design and implementation of IT systems. From the viewpoint of architectural activity, software architecture covers the steps necessary to design and implement architecture. With regard to the structural aspect of architecture, software architecture describes the structures of IT systems. From this point on, the terms “IT system” and “system” are used synonymously provided no explicit differentiation is necessary. A system is a unit that consists of integrated software and hardware building blocks and exists for the purpose of fulfilling a functional objective. To achieve this objective, it communicates with its environment and must take account of the conditions defined by the environment.

# http://www.home.zonnet.nl/azwegers/thesis/figures/2_2.gif



# 2. ARCHITECTURE OBJECTIVES

* **To manage complexity**: An architectural model allows one to present the essence of a complex system in a (simple) model. An architectural model supports the ability to comprehend complex systems; it presents them at a level of abstraction at which a system's high-level design can be understood. It supports the analysis of relationships as an aid to understand complexities in a design environment. In particular, an architecture is needed in complex, dynamic environments (Van Waes, 1991). Zachman states that the increased scope of design and levels of complexity of system implementations are forcing the use of architectural models for defining and controlling the interfaces and the integration of the system components (Zachman, 1987). Architectural models abstract away from details instead of from the essential complexity. Brooks claims that 'the complexity of software is an essential property, not an accidental one' (Brooks, 1995; p. 183). Descriptions of a software entity that abstract away its complexity often abstract away its essence.
* **To serve as a set of specifications**: An architecture may be seen as a result of the design process. It is laid down in specifications, which are derived from the requirements, and from which the desired system can be built. Specifying an architecture is concerned with the specification of components, their interactions, and the constraints on these entities and their interactions. These unambiguous specifications define the scope of future development activities, and serve as a basis for further design and implementation activities.
* **Means of communication**: Furthermore, an architectural model may play the role of a means of communication during a system (re-)design process. The architect can use it to visualise various aspects of the system to be designed, thus providing the various parties concerned with a basis for discussion and decision-making. By producing order in chaos, architectural models help each party to clarify its perception of the problem. Visualisation and explanation of the relevant aspects of the problem area, and the possible relationships between them, supports the various actors to focus their attention on the essential elements, thus providing a basis for discussion of the problems.
* **To indicate the most vital system elements**: Furthermore, the architecture determines the nature and quality of a system. As such, an architectural model indicates the invariant or most vital system elements, which must be treated carefully during system re-design. Systems evolve and are adapted to new uses, just as buildings change over time and are adapted to new uses. One frequently accompanying property of evolution is an increasing brittleness of the system, caused by violations of the architecture. Violations of the architecture frequently lead to an increase in problems in the system and contribute to an increasing resistance to change, or at least to changing gracefully.
* **Means to reduce the impact of changes**: Another role of an architecture involves its contribution to the effective re-design of a system. The architecture should reduce the impact of changes to the lower component levels, and to as few components as possible. Both for shop floor control systems and for products, it is advantageous to use as many parts of the existing system or product design as possible. In a re-engineering trajectory, an architectural model of the system allows one to pinpoint and discuss the areas requiring major change, and to integrate the new specifications into the existing model. Furthermore, architectural change is not so much determined by the system components, as well by the interfaces between these components; the ease with which components can be modified, replaced, or with which the system can be extended by new components is dependent on the extent to which the interfaces of the new components match those of the old ones.
* **Means to gain strategic benefits**: Finally,(product) architecture may have certain strategic importance for a company. The development of a new product brings together a wide range of technologies. Only a few of these technologies contribute to ultimate competitive advantage. Successful companies do not compete on (and even give away) the enabling technologies on which their core utility is based. By the architectural design of functions that can be filled in by cheap, standard components, companies profit from the strong competition in the markets for these components, and are free to focus on their true sources of competitive value. In addition, a company might extend the value of its product by publishing the product's interfaces to the outside world. Other enterprises might use this product as an indispensable part for their own products

# 3. SYSTEM DESIGN SPECIFICATION

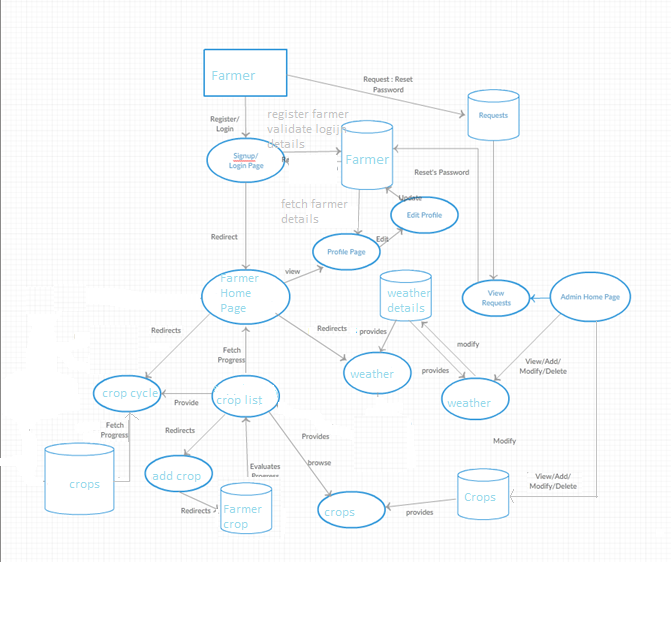
A modular architecture may naturally result in a layered architecture; modules are assigned to specific layers. Layers reflect design decisions based on allowable relations and interfacing constraints. The layers in an architecture represent allowable interfaces among modules. Modules within a layer can communicate with each other. Modules in different layers can communicate with each other only if their respective layers are adjacent (Soni et al., 1995). A layer builds on its underlying layer, which at its turn builds on its underlying layer as well. Consequently, a layer explicitly uses the functionality of its underlying layer, and implicitly uses the functionality of all layers underneath its underlying layer.

Layers are used mainly to solve mapping problems. The mapping task is decomposed in layers: each layer performs a specific part of the mapping. In this sense, the division in layers is part of an architecture. The advantage of layers is the flexibility: changes can be made inside a layer without affecting other layers. A disadvantage of a layered architecture is its rigidity: new layers are hard to be shoved in between existing layers, since this requires a (major) change of interfaces. Examples of the application of layers in mappings are:

* the targets of an enterprise must be mapped on its physical processes; therefore, a strategical, tactical, and operational layer are distinguished;
* data from a database must be mapped on computer screens; therefore, an internal, conceptual, and external layer are distinguished.



**FARMER:**



|  |  |
| --- | --- |
| **Layer-1** | **User Interfaces** |
| **Purpose** | This the layer that the users will use to interact with the system. |
| **Related Components** | User Interfaces. |
| **Software Interfaces** | Layer 3 and Layer 4 Interfaces |
| **Composition Style** | **Generalization** |
| **Communication Pattern** | **Vertical** |
| **Implementation Steps** | 1. Create App 2. Host Server 3. Access database to present profile to user 4. Present farmers information 5. Present application information |

|  |  |
| --- | --- |
| **Layer-2** | **Data Processing** |
| **Purpose** | This Layer processes the data of the app. |
| **Related Components** | Goal 2 and Goal 4 components |
| **Software Interfaces** | Layer 1 and Layer 4 Interfaces |
| **Composition Style** | **Aggregation** |
| **Communication Pattern** | **Horizontal** |
| **Implementation Steps** | 1. Fetch Data 2. Store data in database 3. Remove duplicate data 4. Perform processing on data 5. Store the result back to the database 6. Fetch the final result |

|  |  |
| --- | --- |
| **Layer-3** | **User Queries Processing** |
| **Purpose** | The User Query Processing is done in this layer of components. |
| **Related Components** | Goal 6 and Goal 7 components |
| **Software Interfaces** | Layer 1 and Layer 4 Interfaces |
| **Composition Style** | **Composition** |
| **Communication Pattern** | **Horizontal** |
| **Implementation Steps** | 1. Accept Query 2. Extract the attributes required by the query 3. Gain Database access 4. Verify Database access 5. Query the required info from database 6. Validate data from the response 7. Store related data to database 8. Farmer can access information from the database |

|  |  |
| --- | --- |
| **Layer-4** | **Data Access** |
| **Purpose** | The Data Access and Acquiring is done in this layer. |
| **Related Components** | Goal 1 and Goal 3 components |
| **Software Interfaces** | Layer 3 and Layer 4 Interfaces |
| **Composition Style** | **Aggregation** |
| **Communication Pattern** | **Vertical** |
| **Implementation Steps** | 1. Fetch data from data sources 2. Validate received data 3. Populate farmer database 4. Populate crops database 5. Populate weather information 6. Build Farmers Profiles |

**T.Y. B. Tech.**

**CS 303: Software Engineering Laboratory**

Assignment No: 10

**ERP Farmers**

**System Review and Acceptance**

***16.04.2109***

***Version 1.0***

|  |  |  |  |
| --- | --- | --- | --- |
| Project Group Information | | | |
| Roll. No. | **Gr. No.** | **Name** | **Roles** |
| 59 | **161641** | **Rohan Bhukne** | **Design and front End** |
| 62 | **161259** | **Aniket Thaware** | **Backend** |
| 73 | **161854** | **Rushabh pahade** | **Leader** |

**Approved By: Dr M. R. Dube**

**Academic Year: 2018-19 Semester: II**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Title** | **Page** |
| 1 | Introduction | **73** |
| 2 | Review Types | **78** |
| 3 | Verification Summary | **80** |
| 4 | Verification Matrix | **83** |

# 1. INTRODUCTION

*At the time of the scheduled peer review, ensure proper representation and preparation by the reviewers. Provide clarifications on the work products. Present comments and listen to the comments of the other reviewers. Comments can be presented either by page or by reviewer. Keep the comment discussions short with a focus on detection, not correction. Editorial comments are provided separately and are not discussed at the scheduled review.*

*Participate in categorizing comments. The comments will be categorized and documented as errors, defects, and action items. Refer to the definitions for the categorization rules, which are summarized as follows:*

* *Errors (i.e., problems in the material currently under peer review).*

*Optionally, errors are subcategorized as major (affects functionality and/or performance) and minor (does not affect functionality and/or performance).*

* *Defects (i.e., problems in materials previously peer reviewed).*

*Optionally, defects are also subcategorized as major and minor.*

*Note: Defects will further be categorized as delivered or undelivered in the program’s change request system.*

* *Action items (i.e., unresolved comments requiring further investigation)*
* *A comment can remain categorized as a comment if the reviewers and presenters agree that there is no error, defect, or action item required.*

*To complete the peer review you must identify errors, defects, and action items to be resolved and documented. If needed, follow the program’s or project’s defined decision-making processes to elevate and reconcile any issues encountered in resolving peer review errors, defects, or action items with appropriate stakeholders. To ensure completion, per- form the following:*

* *Correct all errors and update the peer review information to indicate that the error is resolved.*
* *Submit change request paperwork for all defects. The status and tracking of the defect corrections are then handled through the change request system. The defects associated with the peer review should indicate this transfer and are categorized as resolved, allowing the peer review to be closed.*
* *Resolve and complete all action items. If any action items cannot be completed within the two-week period, these action items should be moved to the program- or project-level action item tracking system. The action items associated with the peer review should indicate this transfer and are categorized as resolved, allowing the peer review to be closed.*

# 2. REVIEW TYPES

*Design and code reviews promise to improve software quality, ensure compliance with standards, and serve as a valuable teaching tool for developers. As with most practices, there are subtle nuances surrounding how they're performed that can dramatically affect their value. In some organizations, reviews are a valuable aspect of the software lifecycle. In others, they are a necessary evil tainted with political bureaucracy and big egos. Suboptimal reviews conducted late in the lifecycle are often misguided due to few objective guidelines that help guide the review process. When used throughout the development lifecycle, code and design quality metrics are valuable inputs to the review process.*

* 1. *Reviews Increase Agility Continuous Integration.*

*Agile practices are abundant, and for many teams interested in increasing their agility, valuable energy and resources have been devoted to improving these practices. Because of this, many teams have abandoned reviews while emphasizing other aspects of agility. But, reviews are an important tool in the agile toolkit.*

*A driving principle of the Agile Manifesto is continuous attention to technical excellence. Another is embracing and harnessing change as an opportunity to increase customer advantage. For developers, change often begins and ends with modifications to the source code. A poorly designed application with smelly code is a breeding ground for risk that makes change incredibly difficult, and is the greatest technical inhibitor to increased agility. Effective reviews that emphasize design quality and code cleanliness are an important aspect of increased agility. Reviews done right help ensure continuous attention to technical excellence. Unfortunately, not all reviews are done right.*

*1.2 Review Worst Practices*

*Some development teams find reviews a healthy and valuable asset to developers and the project team. Other teams realize little value from their review process. There are numerous causes for painful and ineffective reviews. Some symptoms of ineffective reviews include:*

* *Witch hunt reviews - Many reviews degrade quickly into attack and defend mode. This often occurs because the developer who wrote the code feels attacked and threatened when reviewers make direct and opinionated statements about the code. Nothing could be less productive.*
* *Curly brace reviews - Some reviews emphasize formatting and comments instead of more serious problems. Is placement of curly braces and misspelled comments really that important? Curly brace reviews are feeding ground for the anal retentive, and provide no real value.*
* *Blind reviews - Often times, reviewers walk into the review meeting having never laid eyes on the code they are about to review. Most of the review time is spent trying to figure out what the code does. Spending time in the review meeting attempting to understand the code instead of reviewing it for more serious ailments is a waste of time.*
* *Exclusionary reviews - Many times, the code provided for the review is only a sampling of the code written. For example, unit tests might be excluded from the review. In an unhealthy review environment, providing impartial and incomplete code listings will leave the reviewers wondering how the code actually works.*
* *Tree killer review - If you can't baffle them by providing half of what they need to understand the code, then maybe overwhelming them by providing thousands of lines of code might work. Waiting until codebase is incredibly large to host the first review is entirely ineffective. Not only is it to difficult to provide effective feedback on a large codebase, these reviews are often held late in the lifecycle and do not allow the developer to improve her code based on the feedback received.*
* *Token review - It's not uncommon for management to dictate that reviews be held. Token reviews are typically held for political reasons. Management wants to ensure that all code is reviewed for auditing purposes. Unfortunately, developers realize very little value surrounding these reviews. Any problems found are not fixed unless they are absolutely critical. Since the primary motivation is an audit trail for management, the team has little motivation to improve the code.*
* *World review- The reviews conducted with great number of people in attendance. This can be incredibly intimidating for the developers whose code is being reviewed, and it is not sure what value it provides to invite so many people. A few developers, up to five, should serve all the needs required of the review process. If more people want to provide input, there are better ways.*

*The Design checklist is as follows:*

* *Deficiencies and conflicts in requirements, architecture, or program/project plans will be reported.*
* *Design decisions and the decision rationales will be recorded according to plans and defined processes.*
* *Top-level software components of the software end item will be identified and described.*
* *Static relationships between top-level software components will be defined.*
* *Dynamic relationships between top-level software components will be defined.*
* *The concepts of execution of the software end item and its components will be defined.*
* *External interfaces of the software end item and its components will be identified and described.*
* *Top-level software components will be decomposed into lower-level software units.*
* *Internal interfaces between software units will be identified and described according to the standards identified by the project.*
* *Design traceability data will be documented according to plans, processes, and product standards.*
* *Design definitions will be documented according to plans, defined processes, and standards.*
* *Measurement and estimated data will be collected.*
* *Applicable work products will be submitted for peer reviews in accordance with project plans.*
* *Applicable work products will be submitted for control in accordance with program or project plans.*

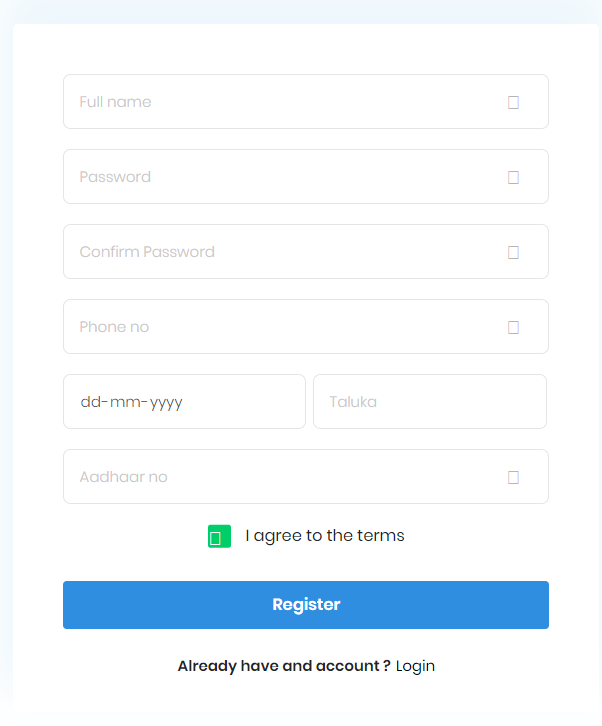
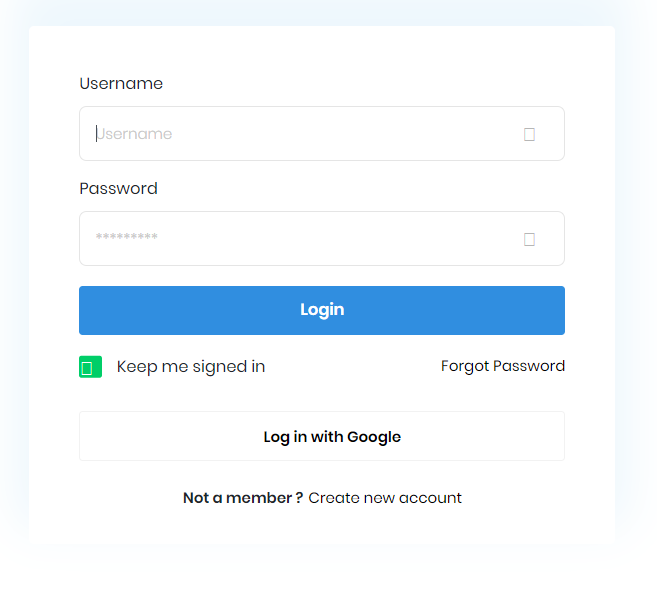
# 3. VERIFICATION SUMMARY

*Note: The verification summary is required to be written for all the objectives and processes as they were detailed as User Stories. Replicate the standard template for objectives and process for the goals.*

## **4. VERIFICATION STEPS: GOAL-1**

|  |  |
| --- | --- |
| **Objective-1** | **Acquire farmers Data** |
| **Purpose** | The purpose is to develop Farmer Profile to enable the system to use that information later. |
| **Target Audience** | farmers |
| **Status** | Completed |
| **Role:** | **As an**end user, developer |
| **Verification Steps** | 1. Verify that farmer profile is created. |
|  | 2. Verify that farmers’ data is collected. |
|  | 3. Verify preliminary database is created. |
|  | 4. Verify that farmer data Spyder is created. |
|  | 5. Verify Database structure. |
|  | 6. Verify that farmers’ database is populated. |
|  | 7. Verify that backup is generated. |
|  | 8. Verify that the backup accessible. |
|  | 9. Verify proper database privileges and security. |
|  | 10. Verify farmers’ profile is launched. |

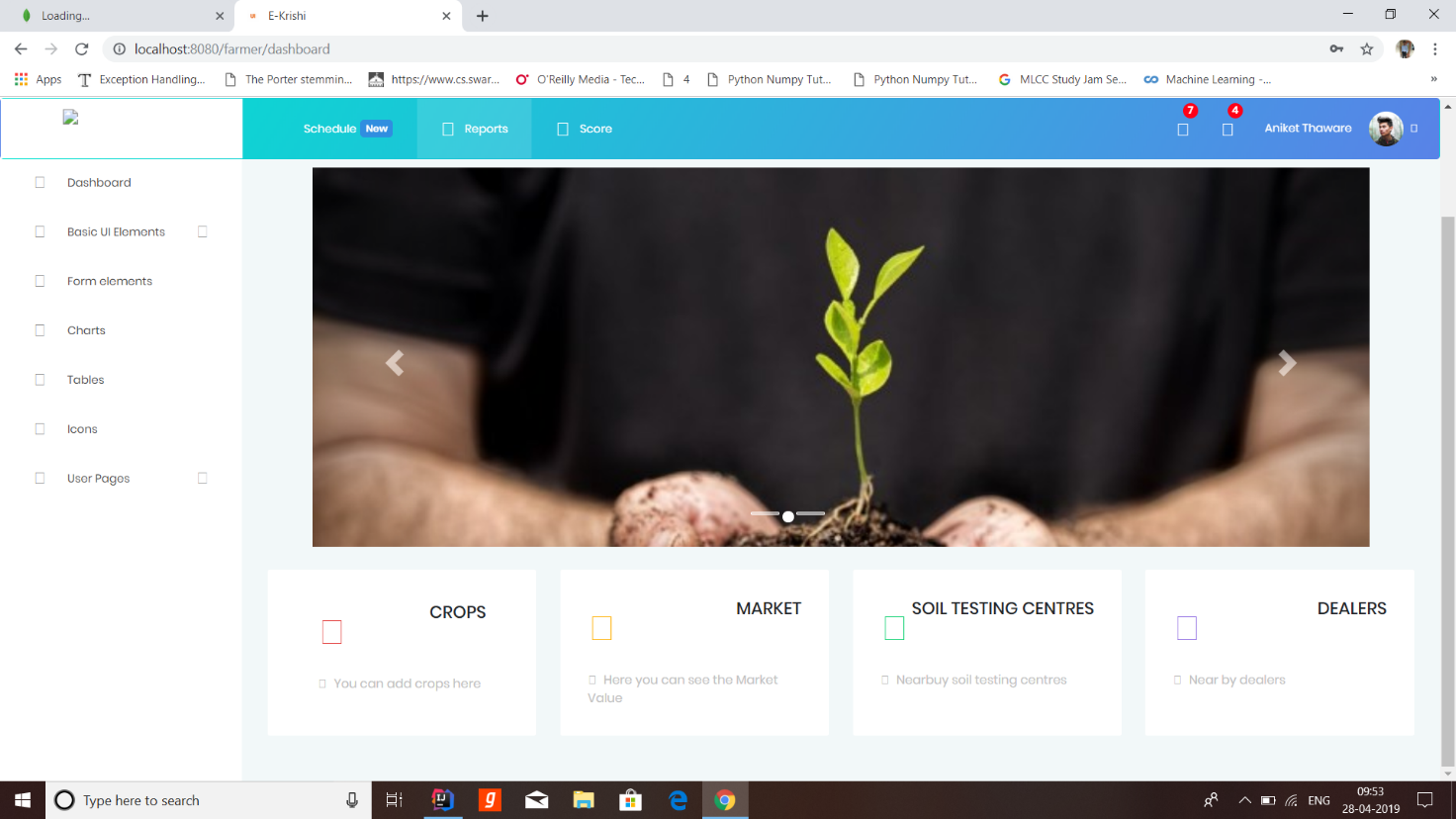
|  |  |
| --- | --- |
| **Objective-2** | **Analyse Farmer’s Data** |
| **Purpose** | To decide the Farmer analysis process of the system and the methodology to follow. |
| **Target Audience** | Internal Stakeholders |
| **Status** | Completed |
| **Role:** | **As a**developer |
| **Verification Steps** | 1. Verify Organisation of database attributes |
|  | 2. Verify the Design patterns for attributes |
|  | 3. Verify the output parameters |
|  | 4. Verify the important parameters priority |
|  | 5. Verify the short-listed attributes |
|  | 6. Verify Organisation of the parameters |
|  | 7. Verify the formulated observations |
|  | 8. Verify correspondence with Analysis team |
|  | 9. Verify the consolidation of analysis process |
|  | 10. Verify the final analysis methodology |



## 5. VERIFICATION STEPS: GOAL-2

|  |  |
| --- | --- |
| **Objective-1** | Create Customer Profile |
| **Purpose** | Make separate database for Customer. |
| **Target Audience** | Customers |
| **Status** | Completed |
| **Role:** | **As a customer** |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1. Request creation of Customer Profile |
|  | 2. Collect Customer data |
|  | 3. Create a preliminary database |
|  | 4. Create a Customer Data Spyder |
|  | 5. Formulate database structure |
|  | 6. Populate Customer Database |
|  | 7. Generate a backup |
|  | 8. Share backup with Project Team |
|  | 9. Assign database privileges |
|  | 10. Launch Customer Profile Page |

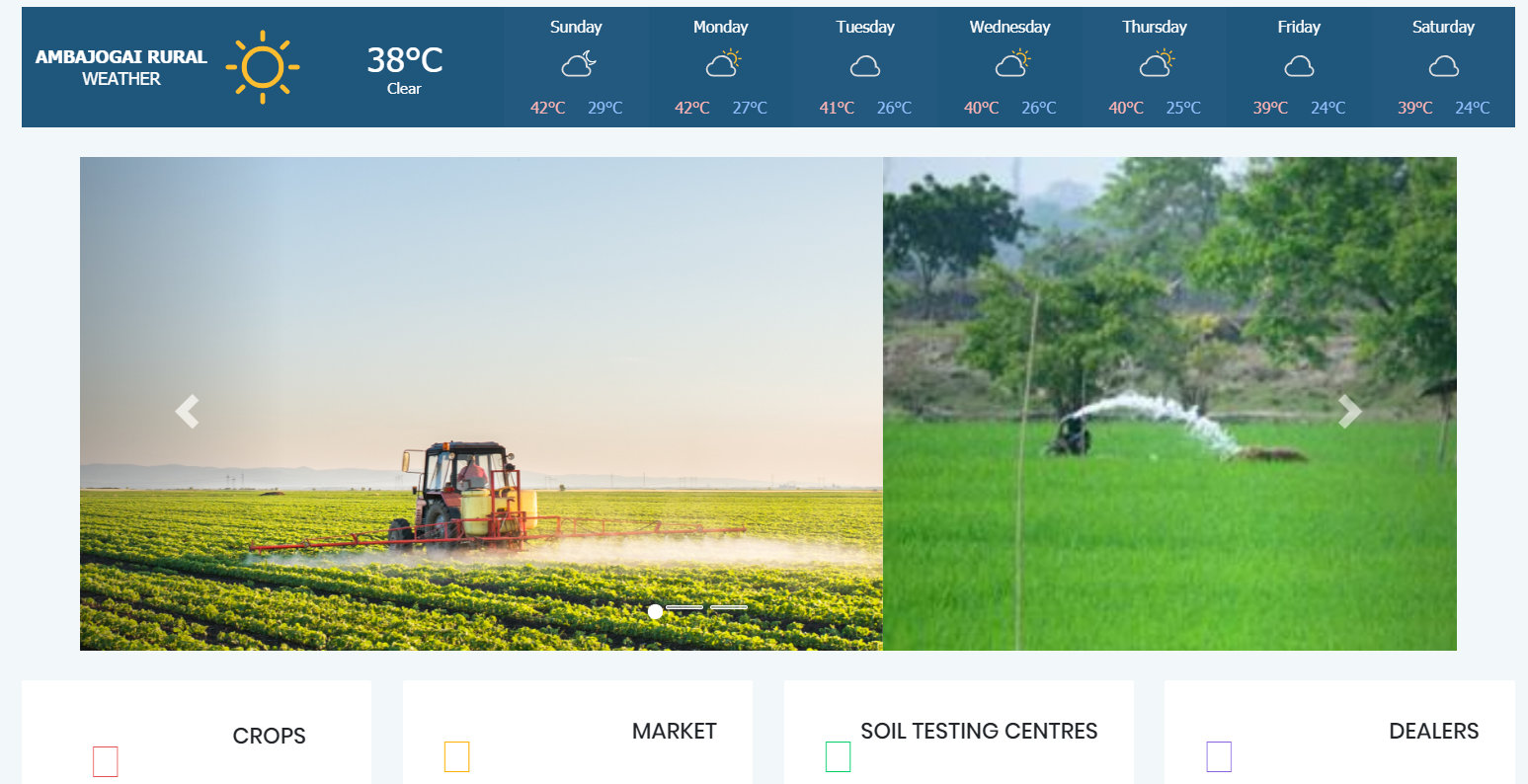
|  |  |
| --- | --- |
| **Objective-2** | Analyse Customer Data |
| **Purpose** | To decide the Customer analysis process of the system and the methodology to follow. |
| **Target Audience** | Customers |
| **Status** | Completed |
| **Role:** | **As a**Developer |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1. Organise database attributes |
|  | 2. Design patterns for attributes |
|  | 3. Conceptualise output parameters |
|  | 4. Prioritise the important parameters |
|  | 5. Draw a map for the shortlisted attributes |
|  | 6. Organise the parameters |
|  | 7. Record formulated observations |
|  | 8. Correspond with Analysis team |
|  | 9. Consolidate outline of analysis process |
|  | 10. Construct final analysis methodology |



**6. VERIFICATION STEPS: GOAL-3**

|  |  |
| --- | --- |
| **Objective-1** | To make farmers profile |
| **Purpose** | To extract customers’s data from database and performed operation on received information. |
| **Target Audience** | Customers and Farmers |
| **Status** | Completed |
| **Role:** | **As a**Developer |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1. Request creation of customers’s profile |
|  | 2. Collect customers’s data |
|  | 3. Create a preliminary database |
|  | 4. Create a customers’s Data Spyder |
|  | 5. Formulate database structure |
|  | 6. Populate Customers’s database |
|  | 7. Generate a backup |
|  | 8. Share backup with Project Team |
|  | 9. Assign database privileges |
|  | 10. Launch Farmer Profile Page |

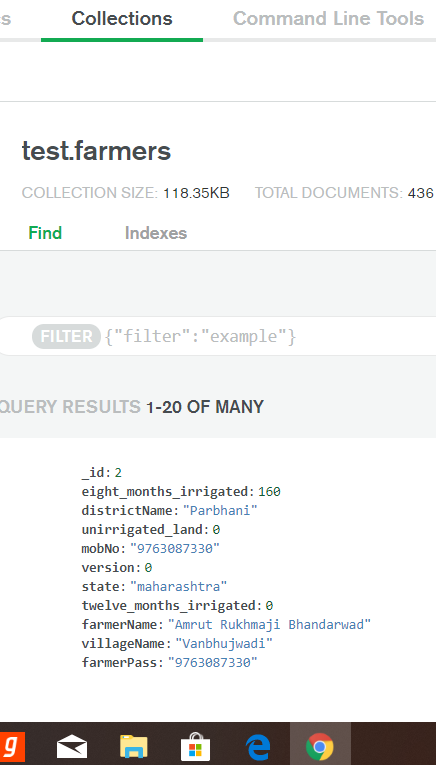
|  |  |
| --- | --- |
| **Objective-2** | **Analyse farmers Data** |
| **Purpose** | To decide the customers analysis process of the system and the methodology to follow. |
| **Target Audience** | Internal Stakeholders |
| **Status** | Completed |
| **Role:** | **As a Developer** |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1. Organise database attributes |
|  | 2. Design patterns for attributes |
|  | 3. Conceptualise output parameters |
|  | 4. Prioritise the important parameters |
|  | 5. Draw a map for the shortlisted attributes |
|  | 6. Organise the parameters |
|  | 7. Record formulated observations |
|  | 8. Correspond with Analysis team |
|  | 9. Consolidate outline of analysis process |
|  | 10. Construct final analysis methodology |



### 7. VERIFICATION STEPS: GOAL-4

|  |  |
| --- | --- |
| **Objective-1** | **Perform Customer related activities.** |
| **Purpose** | Customers will receive the crop details. |
| **Target Audience** | Customers |
| **Status** | Completed |
| **Role:** | **As a customers** |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1. Register all customers to this app |
|  | 2. Retrieve Crop advertisement. |
|  | 3. Retrieve notifications. |
|  | 4. Submit feedback to Farmer |
|  | 5. Provide prices to the customer of advertisement. |
|  | 6. Get notifications. |
|  | 7. Get category wise advertisement. |
|  | 8. Buy from any Category |
|  | 9. Ask doubts to Farmer for any doubt |

|  |  |
| --- | --- |
| **Objective-2** | **Classify Customers** |
| **Purpose** | To classify customers |
| **Target Audience** | Customers |
| **Status** | Completed |
| **Role:** | **As a** Developer |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1.Get customers’s attributes and data |
|  | 2.Examine data distribution per attribute |
|  | 3.Fix inconsistencies in selected attributes |
|  | 4.Prototype groups according to attributes |
|  | 5.Receive subject wise content |
|  | 6.Access study material groups |
|  | 7.Form a basis/gist for the group |
|  | 8.Choose encoding method for the basis |
|  | 9.Establish results of encoding analysis |
|  | 10.Integrate results |



### 8. VERIFICATION STEPS: GOAL-5

|  |  |
| --- | --- |
| **Objective-1** | Display weather information |
| **Purpose** | To make weather information effective. |
| **Target Audience** | Famer |
| **Status** | On-going |
| **Role:** | **As a** *developer* |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1. Upload location wise weather |
|  | 2. Implement AI module |
|  | 3. Provide input to search |
|  | 4. Implement level wise search |
|  | 5. Implement level wise search |
|  | 6. Record weather details |
|  | 7. Optimize search time |
|  | 8. Record most frequent search queries |

|  |  |
| --- | --- |
| **Objective-2** | Categorize weather information |
| **Purpose** | To display weather animation to farmer |
| **Target Audience** | farmer |
| **Status** | On-going |
| **Role:** | **As a** *Developer* |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1.Get search attributes and data |
|  | 2.Examine data distribution per attribute |
|  | 3.Fix inconsistencies in selected attributes |
|  | 4.Prototype groups according to attributes |
|  | 5.Receive subject wise content |
|  | 6.Access location wise weather |
|  | 7.Form a basis/gist for the group |
|  | 8.Choose encoding method for the basis |
|  | 9.Establish results of encoding analysis |
|  | 10.Integrate results |

### 6 VERIFICATION STEPS: GOAL-6

|  |  |
| --- | --- |
| **Objective-1** | To show progress of seeded crop. |
| **Purpose** | To display crop cycle |
| **Target Audience** | Famers and customers |
| **Status** | Completed |
| **Role:** | **As a** *farmer* |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1.Prepare each step of growing crop. |
|  | 2.Get topic’s data from respected source |
|  | 3. Implement time clock |
|  | 4.Create database for all crops. |
|  | 5.Create database for uploaded crop by farmer |
|  | 6. Retrieve crop submitted by customers |
|  | 7. Archive all crops which are not selected |
|  | 8. Send feedback to system |

|  |  |
| --- | --- |
| **Objective-2** | To display crop cycle according to crop type. |
| **Purpose** | To display crop cycle |
| **Target Audience** | Farmers. |
| **Status** | Completed |
| **Role:** | **As a**farmers |
| **Verification Steps** | **I want to** *<perform some task>* |
|  | 1. Receive crop cycle from registered crop by Farmer |
|  | 2. Farmers login through its credentials |
|  | 3. Farmers can get indication for each step of growing |
|  | 4. Farmers can get information for each step separately. |
|  | 5. Time clock should be visible to Farmers |
|  | 6. Farmers can submit crop feedback |
|  |  |
|  |  |

# VERIFICATION MATRIX

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User Story | Step-1 | Step-2 | Step-3 | Step-4 | Step-5 | Step-6 | Step-7 | Step-8 | Step-9 | Step-10 |
| G1:O1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G1:P1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G1:P2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G1:O2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G1:P1 | √ | √ | x | x | √ | x | √ | √ | √ | √ |
| G1:P2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G2:O1 | √ | √ | √ | √ | √ | √ | x | √ | √ | √ |
| G2:P1 | √ | √ | x | x | x | x | √ | √ | √ | √ |
| G2:P2 | √ | √ | √ | √ | √ | √ | √ | x | √ | √ |
| G2:O2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G2:P1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G2:P2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G3:O1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G3:P1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G3:P2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G3:O2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G3:P1 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G3:P2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G4:O1 | √ | √ | √ | x | √ | √ | √ | √ | √ | √ |
| G4:P1 | √ | √ | √ | √ | √ | x | √ | √ | √ | √ |
| G4:P2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G4:O2 | √ | √ | √ | √ | √ | x | √ | √ | √ | √ |
| G4:P1 | √ | √ | √ | √ | √ | √ | √ | x | √ | √ |
| G4:P2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G5:O1 | x | x | x | √ | √ | √ | x | x | √ | √ |
| G5:P1 | √ | √ | x | x | x | √ | √ | √ | x | x |
| G5:P2 | √ | √ | √ | √ | x | x | x | x | x | x |
| G5:O2 | x | √ | √ | x | x | x | x | x | √ | √ |
| G5:P1 | √ | √ | √ | √ | √ | √ | √ | √ | x | x |
| G5:P2 | √ | √ | √ | √ | √ | √ | x | √ | √ | √ |
| G6:O1 | √ | √ | x | √ | √ | √ | √ | √ | √ | √ |
| G6:P1 | √ | √ | √ | √ | √ | √ | x | x | √ | √ |
| G6:P2 | √ | √ | √ | x | x | x | √ | √ | √ | √ |
| G6:O2 | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| G6:P1 | √ | √ | √ | √ | √ | x | √ | √ | √ | x |
| G6:P2 | √ | √ | √ | x | √ | √ | √ | x | x | X |
| G7:O1 | X | X | X | X | X | X | X | X | X | X |
| G7:P1 | X | X | X | X | X | X | X | X | X | X |
| G7:P2 | X | X | X | X | X | X | X | X | X | X |
| G7:O2 | X | X | X | X | X | X | X | X | X | X |
| G7:P1 | X | X | X | X | X | X | X | X | X | X |
| G7:P2 | X | X | X | X | X | X | X | X | X | X |