**“INSERTION SORTING APP”**

**ANDROID APPLICATION DEVELOPMENT**

**HOMEWORK 1**

**TEAM 7**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email** | **CWID** |
| YAMINI GADIDAM | [yaminigds@csu.fullerton.edu](mailto:yaminigds@csu.fullerton.edu) | 805317559 |
| SHIVAM RANA | [shivamrana@csu.fullerton.edu](mailto:shivamrana@csu.fullerton.edu) | 805613635 |
| SREE HARSHA RAVILLA | [sreeharsha@csu.fullerton.edu](mailto:sreeharsha@csu.fullerton.edu) | 805615325 |
| ASHISH PAUL STEPHEN | ashish\_paul94@csu.fullerton.edu | 805663515 |

**SUBMISSION DATE**

**09/28/2016**

**CPSC 544**

**ADVANCED SOFTWARE PROCESS**

**DR. CHANG-HYUN JO**



**REVISION HISTORY**

|  |  |  |
| --- | --- | --- |
| **Serial No** | **WORK TITLE** | **Page No** |
| 1. | PROCESS DEFINED FOR THE PROJECT | 3-6 |
| 2. | WORK PRODUCTS, ROLES AND PRACTICES | 7-10 |
| 3. | PRE GAME PLANNING AND STAGING | 11-13 |
| 4. | REQUIREMENTS | 14-15 |
| 5. | ARCHITECTURALLY INFLUENTIAL FACTORS | 16 |
| 6. | USER STORIES | 17-18 |
| 7. | PRODUCT BACKLOG AND ESTIMATES | 19 |
| 8. | ARCHITECTURAL SPIKE | 20 |
| 9. | RELEASE PLAN | 21-22 |
| 10. | TECHNOLOGY PREPARATION | 23-24 |
| 11. | DEVELOPMENT Iteration 1 Iteration 2 Iteration 3 | 25-45 |
| 12. | ACCEPTANCE TESTING | 46-47 |
| 13. | RELEASE | 48 |
| 14. | OTHER ACTIVITIES | 49-52 |
| 15. | LESSONS LEARNED | 53-54 |
| 16. | REFERENCES | 55 |
| 17. | TEAM CHARTER | 56-59 |
| 18. | TEAM EVALUATION | 60 |

1. **PROCESS DEFINED FOR THE PROJECT**

In this project, we used both Scrum and Extreme Programming Agile methodologies to developed our own process. We tried to combine both the process to develop well-defined and useful process. In this process we mainly focused on the development areas on Extreme programming(XP) and the management areas on scrum.

In this process we began to develop our own process which was used to make decisions on what practices are to be used and which are to be discarded. As the phases are to be defined we decided to follow Extreme programming(XP) as it best fits our project.

We defined the team roles we choose the scrum roles which best defines and supports our project. These practices and activities are chosen from both XP and scrum process together.

We chose XP phases instead of scrum phases was mainly because the XP phases are more concern about engineering rather than management and that we needed for our project phases. The phases of the project are defined below:

* **Planning**

In the XP programming life cycle planning is the first phase. In this phase the team creates ‘user stories’ and requirements. Then the product backlog and estimations are created by the team. Then the product backlog is divide into iterations which covers the small part of the functionality or features that are required. The combination of iterations provides the final functional product. In this phase the team must have sprint planning meeting and sprint backlog.

* **Managing**

In this phase, we find an area where all the team members can meet without missing the meeting. We also measure the amount of work that is being done for each iteration and estimations are updates if needed. In this phase we set a sustainable pace to plan our releases and iterations that keep us from getting into project loss. If something goes wrong, we improve our process in order to successfully manage our process.

* **Designing**

In the designing phase team creates a system metaphor and agree on uniform styles and formats to ensure compatibility among the work of different team members.

The guiding principles of this stage are:

* Thrust on simplicity by expressing a thing only once and not adding functionality in anticipation.
* Using systems metaphor or standards on names, class names and methods, and agreeing on uniform styles and formats to ensure compatibility among the work of different team members
* Using Software Class Responsibilities and Collaboration (CRC) Cards that allow for a departure from the traditional procedural mindset and make possible object oriented technology. Such cards allow all members of the project team to contribute ideas, and collate the best ideas into the design
* Creating spike solutions or simple programs that explore potential solutions for a specific problem, ignoring all other concerns, to mitigate risk
* **Coding**

In this phase the programmers develop the code based on the agreed metaphors and standards. They give priority to actual coding over all other tasks when compared to documentation to ensure that the outcome received is valuable at the end of the day. They start to generate codes at the very beginning to have something that shows his/her progress

* **Testing**

In this phase the team uses Unit Tests which are automated tests, and the programmer will write as many as test cases to eliminate errors. Extreme program integrates testing with the development phase rather than at the end of the development phase. All codes have unit tests to eliminate bugs, and the code passes all such unit tests before release.

These are the ‘Scrum processes’ that were used for our project:

* **Sprints and Sprint planning meetings: -** Sprints are iterations of approximately a month or less. In our case we had three iterations and of 10 days each. A Sprint is started by a Sprint planning meeting, where the goal of the Sprint is defined and the Sprint backlog is started by the team members
* **Daily Scrums.**

These are daily online meetings where the team used to discusses what work is currently in progress, and also we take care of any  [obstacle](https://www.google.com/search?biw=1536&bih=792&q=define+obstacle&sa=X&ved=0ahUKEwjGgYLWiKHPAhUP6GMKHbdeCHUQ_SoIIDAA), and also to improve the quality and outcome of the project.

* **Sprint backlog.**

The Sprint backlog is the list of work that's currently assigned to the Sprint, and the amount of work remaining to complete an item of work. It's owned by the project team and is updated on a daily basis. Our sprint backlog was the work assigned every week based on the list of items to be done and completed on time.

The ‘roles of our project’ were chosen from scrum roles. The roles are as follow:

* **Product owner**

Product is the person who is responsible for creating and prioritizing the product backlog, choosing the goals for each sprint and review the work with the team and the stakeholders to continuously improve the quality.

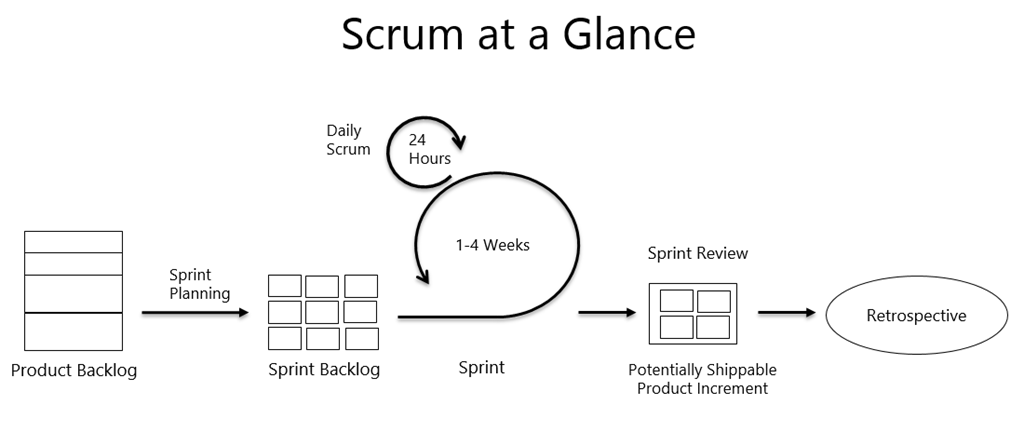
* **Scrum Master**

The Scrum Master is the unfortunate individual who has been charged with removing any impediments that the team may have. Our team Scrum master has helped everyone in every possible manner just like a coach facilitating meetings, and to ensure an excellent product backlog, was constantly working with the product owner and was always ready for the next sprint.

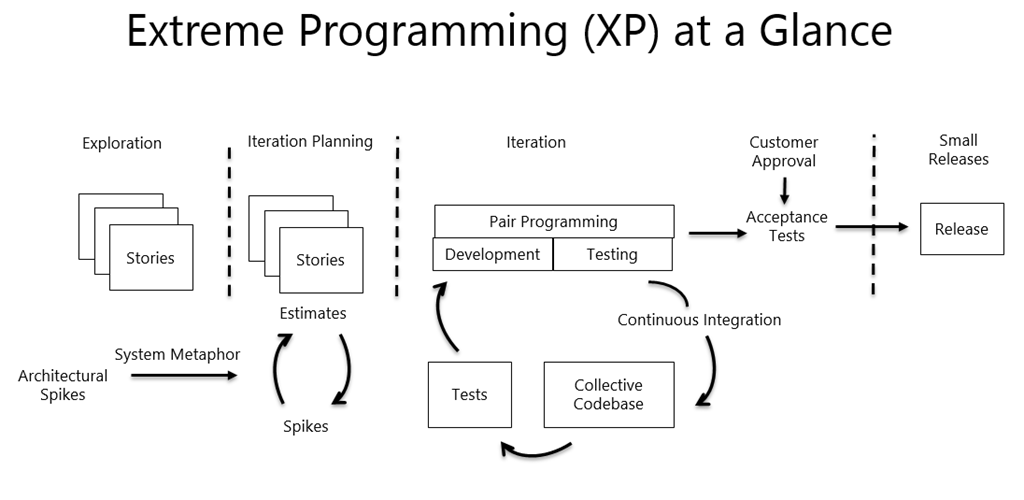
It’s the job of the scrum master to ensure that each person in the team is following the process and help the team members to move forward and be [fruitful](https://www.google.com/search?biw=1536&bih=792&q=define+fruitful&sa=X&sqi=2&ved=0ahUKEwiKoMDgi6HPAhUQ82MKHShdA9wQ_SoIHzAA).

* **Development team**

It is the development team whois responsible for building the product. They deliver shippable increments of product at the end of each Sprint to end up with the final product.

****

**Figure 1: Scrum work product and practices.**

****

**Figure 2: Extreme Programming work product and practices.**

**2.WORK PRODUCTS, ROLES AND PRACTICES**

**Work Products and Practices**

* **Product backlog**:

The product backlog lists all the requirements of product which is being developed. It said to be the master list of all the functionalities desired in the product, and each item in the product backlog has its description, a priority and an estimate of the effort needed to complete it. All these ordered list for product will come from the product owner, team members, or stakeholders.

* **Sprint**:

A sprint is one iteration which goes for 30 days. Here only the product owner has the authority to cancel the sprint. Many of them (Sutherland and vodde) predicted that a sprint goes long for 2-6 weeks.

* **Sprint planning**:

Each sprint begins with the sprint planning. Here basically the scrum team selects and understands the work to be done in the sprint. Usually the time recommended for sprint planning meeting is around two hours in a week of the sprint duration. since the meeting is sprint planning.

* **Sprint backlog**:

Sprint backlog tells us about the detailed plan for development during the next sprint. The sprint backlog is the list of refined product backlog items chosen for development in the current sprint. The team displays its plans and progress so that all team members and stakeholders can always see what the team is accomplishing.

* **Release plan**:

Release plan specifies the goal of the release, greatest priority items in the Product Backlog, the major risks, and also the overall features and functionality that the release will have. It takes care of the probable delivery date and cost.

* **Daily scrum**: Daily scrum briefs about having meetings every week. We used to meet at a same place and time according to schedule The meeting should be short and not more than for a maximum of 15 minutes. During the meeting, each development team member gives three bits of information:

1. What has been accomplished by each member since the last Daily Scrum.
2. What plans need to accomplish between now and the next Daily Scrum.
3. What is impeding progress.

The Daily Scrum is not a reporting event. It's a communication meeting within the development team that helps ensure that all team members are on the same page and moving forward. The Daily Scrum leads to transparency, trust, and better performance.

* **Sprint review**:

At the end of each sprint, the Scrum team and stakeholders review the resulting product increment. This meeting is called a sprint review and should be time boxed one hour per week of the sprint. So if the sprint lasts two weeks, the recommended timebox for the sprint review is two hours. During the meeting, the team members look at where they are and collaborate on how they might move forward. Everyone has input at the sprint review. And naturally, the product owner makes the final decisions about the future, updating the product backlog as appropriate.

Some common components of the meeting include:

1. An overview of the product increment.
2. A demonstration of the product increment.
3. A discussion of what team members observed during the sprint, or perhaps product ideas that came to mind.
4. A discussion about the state of the product backlog, possible completion dates, and what might be done by those dates.
5. An update of the product backlog.

* **Product increment**: Product increment is an integrated, shippable subset of the product. It is the most important Scrum artifact. A product increment is the "goal line" for each sprint. Additional indicators of progress: burn-down charts and task boards.
* **Product backlog refinement**: Because of this fluid environment, product backlog refinement is an ongoing activity throughout a Scrum project. When you refine the product backlog, you:

1. Confirm the order of the product backlog items.
2. Remove or demote items that no longer seem important.
3. Add or promote items that come up or become more important.
4. Split larger items into smaller items.
5. Merge smaller items into larger items.
6. Estimate items.
7. Identify which items are sprint-ready.

During this process, you give special attention to selecting items coming up for the next sprint. Things to consider include:

1. Each item for the sprint should represent an increment of "business value."
2. The development team needs to be able to build each item within a single sprint.
3. Both the stakeholders and the entire Scrum team need to be clear on what is intended.

**Roles**

* **Product owner**: The product owner holds the vision for the product. They facilitate communication between the team and the stakeholders and ensure the team is building the right product. They describe what should be built and why, but not how.  
  To fulfill the role, the product owner:

1. Decides what goes into the product backlog and, equally important, what does not.
2. Maintains the product backlog and orders the items in the backlog to deliver the highest value.
3. Works with the team and the stakeholders to continuously improve the quality of the product backlog and everyone’s understanding of the items it contains.
4. Decides which product backlog items to ask the team deliver in the current sprint.
5. Decides when to ship the product, with a preference toward more frequent delivery.

The product owner may be supported by other individuals but must be a single person to maintain clarity of the vision and priorities.

* **Scrum master**: Scrum master helps the team best use Scrum to build the product, keeps the Scrum team productive and learning, must have a good understanding of the Scrum framework and ability to train others.

1. Coach the team.

2. Keep the team moving forward.

3. Help everyone understand Scrum.

* **Development team**: Development team builds the product.

1. Cross-functional group of professionals.

2. Deliver each increment of the product.

**3. PRE-GAME: PLANNING AND STAGING**

1. **Vision**

The vision of our application is to perform the ‘Insertion Sorting’, when an input is given the application on pressing the sort button should sort the given input and provide us each step of sorting and the sorted array in the output. The supplication should quit when Exit button is pressed.

1. **Description**

The Insertion sorting application is an Android application. Android is a mobile based operating system which is developed by Google based on Android open source project(ASOP). It is basically used in smartphones and tablets. As Android is an open source application which allow developers to develop applications. Applications are written are written in java programming language and run on Android studio. Which is used for creating third party android applications.

Our Insertion sorting application is an Android application which can be run on any Android device. This application is used to sort the given integer values it sorts the integers from 0 to 9 when a minimum of 2 integer or a maximum of 8 integer array is given as input. When sort button is pressed it generates all the sorted array steps and clears the input area when the clear button is pressed. The application swill quit when the Exit button is pressed.

1. **Goal of the Project**

* Develop a simple application which is fully functional and user friendly.
* The application should adapt to any requirements which may be functional or non-functional requirements.
* Plan for multiple releases.
* Make sure it is adaptable to any changes in the future.
* Build the application by using appropriate software methodology.

1. **Goals of the organization/Business**

* Build a Solid business relationship
* Reliable software application
* Improve customer satisfaction and increased productivity
* Receive a good customer feedback
* Balanced growth
* Reduce operational costs
* Create criteria for measuring the company’s advancement
* Refine the targets with usage and estimating systems
* Expand into new locations

1. **System features**

* Show error messages when user enter invalid input.
* The insertion sort array is displayed
* Programming sequence

Enter array of integers from (0-9)

Input size should be min of 2 max of 8

Entered array is to be sorted

If entered values less than array size of 2 or more than 8 error is displayed

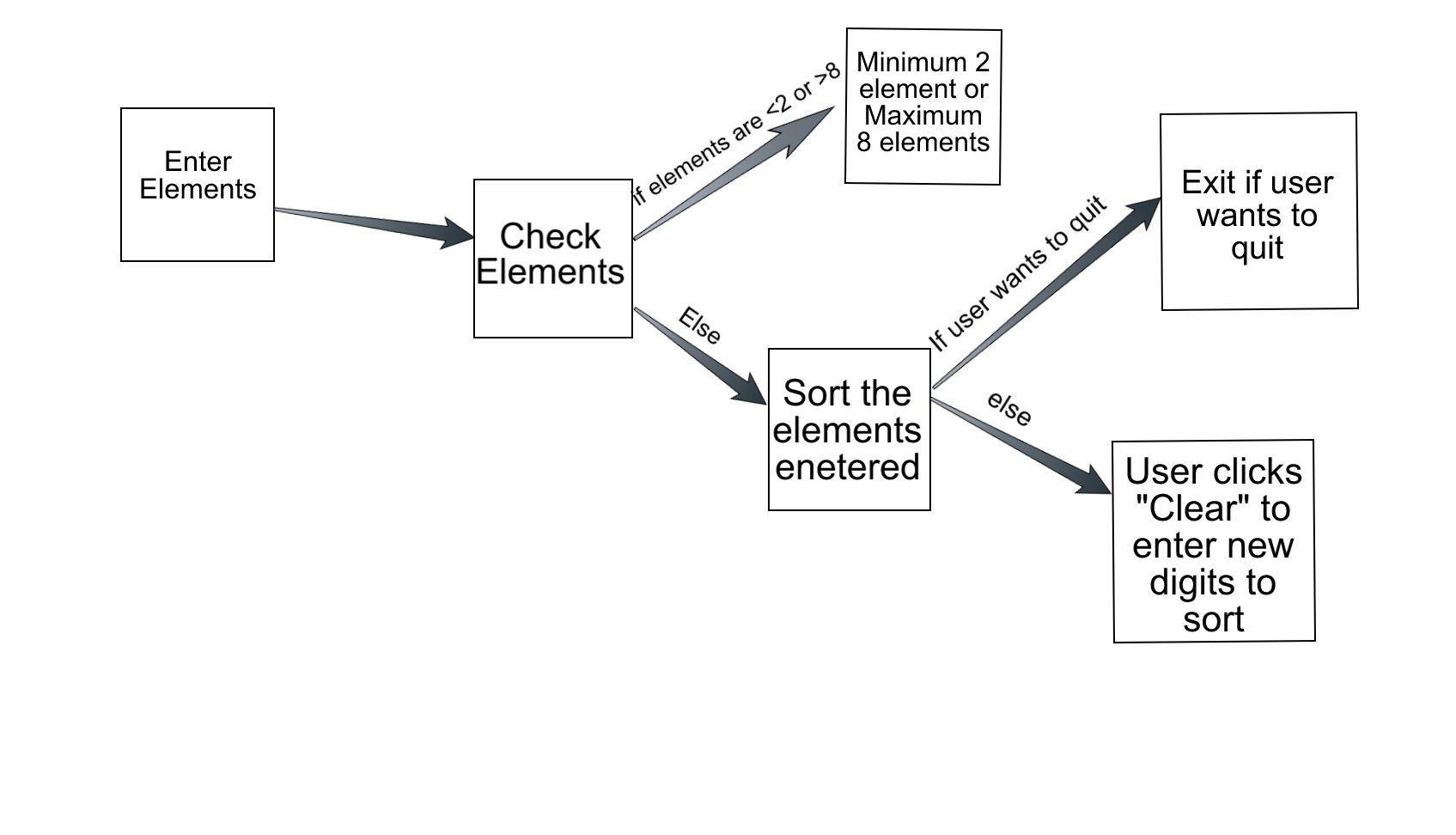
The complete sorted array with steps is displayed when ‘Sort button’ is pressed

The application is closed when pressed ‘Exit’

1. **System context diagram**

“System context diagram defines the boundary between the system, or the part of the system, and its environment, showing the entities that interact with it”. In our Insertion sorting application, the system (the core of the application) is connected to external entities, such as user interface. The user interface on behalf of user, requests the system to perform sorting or to exit the application. The system responds back to user interface with appropriate response.

Fig. System Context Diagram



1. **Stake holders**

There are four types of stake holders

* Developers: They construct and deploy from given requirements
* Users: They define the systems functionality and make use of it
* Testers: They test the system and ensure it is suitable foe use
* Manager: coordinates all the activities

**4.REQUIREMENTS:**

**Functional requirements:**

* **R1:** 3 buttons

The user interface has 3 buttons. The first button is “sort” used for sorting the integers entered by the user. The second button is “clear”, which is used for clearing the input given by the user. The last one is “exit”, used to quit the application.

* **R2**: 1 Input box:

The UI has 1 textbox where the user has to enter the inputs. It accepts minimum of 2 integers and maximum of 8, where the value ranges from 0 to 9.

* **R3:** Output:

The output provides the sorted array of integers given by the user.

* **R4:** Input must be an integer number.

The sorting application only accepts integer values ranging from 0 to 9.

* **R5:** Show error message for invalid input.

When the user inserts the values greater than 8 integers, then an error is generated as it exceeds the array size.

* **R6:** Show error message for invalid input.

When the user inserts the value less than 2 integers, an error message is generated as the requirement is to enter minimum of 2 integers.

* **R7:** Sorting

Each sorting step is generated and displayed to the user on how the sorting took place.

**Non-functional requirements:**

* **NFR 1:** the application must comply with android OS

Our Sorting Application must be an android application that complies to the android operating system.

* **NFR 2**: performance

Keep the application simple with no additional graphics, which allows the application to respond quickly to users.

* **NFR3:** usability

The system must be simple and minimalistic and user-friendly to navigate through out the application. Also, display error messages so that, the users know where exactly did they go wrong.

* **NFR4:** reliability

The application should not crash when an invalid input is given. It has to continue to perform, by displaying error message.

* **NFR 5:** Availability

The application should be available for the end-users 24/7.

**5.Architecturally Influential Factors**

The factors that influence our sorting application architecture design

* **Android Environment**

As the application should be developed in android environment. The developer should be aware of the android environment and java programming as this is an important factor which might influence our architecture as the application should be suitable for android environment.

* **Time**

Time is a major factor for any project as if the project gets delayed there is a lot of loss financially and it will affect the whole development cycle of the project.

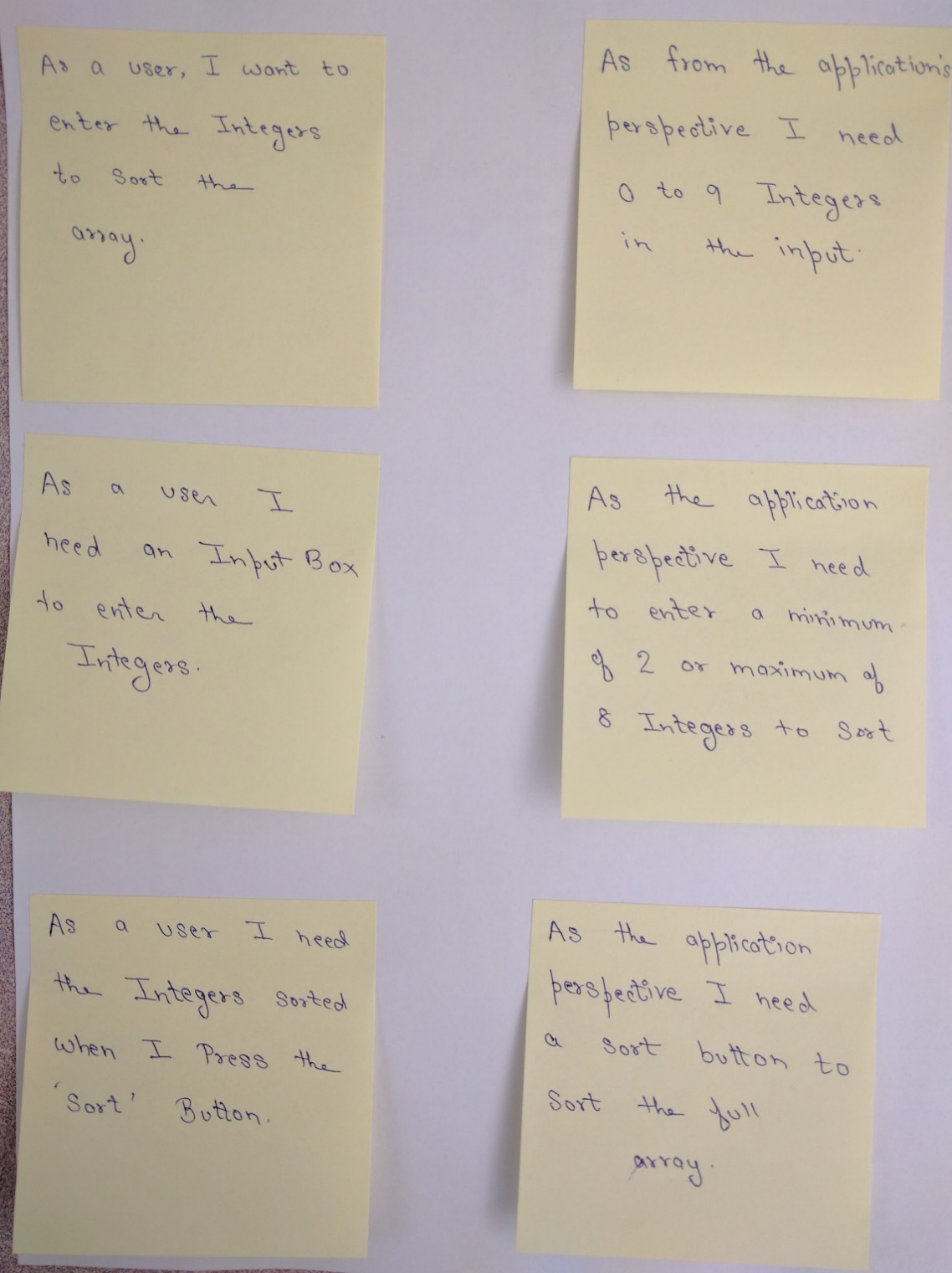
* **Team expertise**

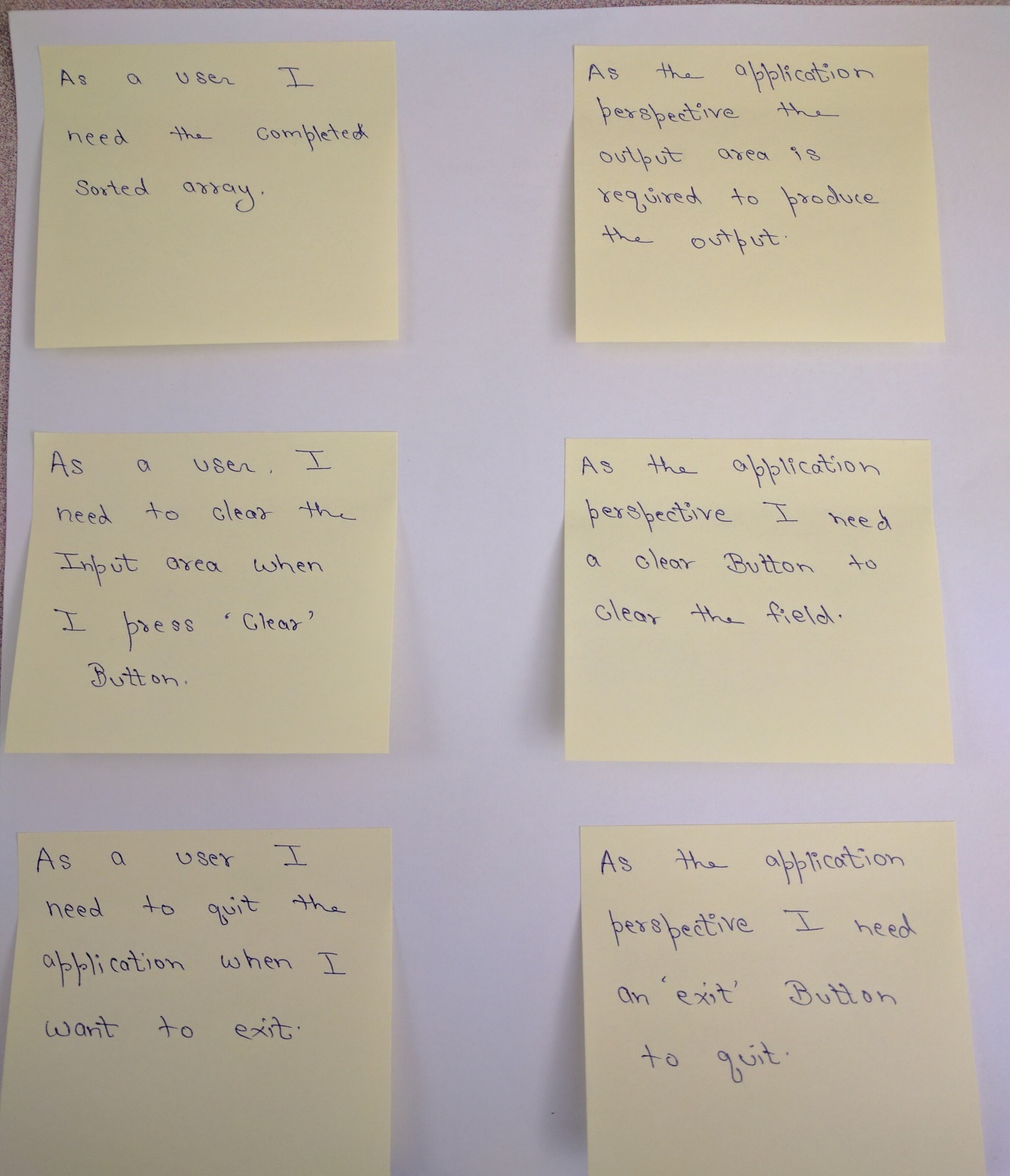
As this is the first time for the team to work in android environment. We need to plan and work accordingly in order to develop the product successfully. In order to achieve we collaborated and met regularly to share the information what we learned individually.

**6.USER STORIES**

User stories describes the features of the application. These are basically hand written cards called as story cards. These cards are used to define perspective of the user and the application.

In the Insertion sorting application, we use these story cards to record the perspectives of the user and the application. The below we show our story cards that are used for our sorting application.

****

****

**7.Product Backlog and Estimates**

The product backlog is the prioritized feature list which contains the description of all functionality desired in the product which includes the user stories, enhancements, attributes and so on. As we are using scrum it is not necessary to start a project with all the full-fledged requirements. In scrum a scrum team and product owner begin by writing down every thing they can they can think for to prioritize the agile process.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Story** | **Number** | **Category** | **status** | **priority** | **Estimate**  **(hours)** |
| Input is an integer number | 1 | Requirement | Not started | High | 1 |
| Input range [0 to 9] | 2 | Constraint | Not started | High | 1 |
| Minimum array size is 2 | 3 | Constraint | Not started | Medium | 0.5 |
| Maximum array size is 8 | 4 | Constraint | Not started | Medium | 0.5 |
| Sort | 9 | Requirement | Not started | high | 1 |
| Show error message | 5 | Enhancement | Not started | High | 2 |
| Display sorted array | 6 | Requirement | Not started | Medium | 1 |
| User friendly GUI | 7 | Requirement | Not started | High | 2 |
| Must be an android application | 8 | Requirement | Not started | High | 7 |

**Product backlog**

**8.ARCHITECTURAL SPIKE**

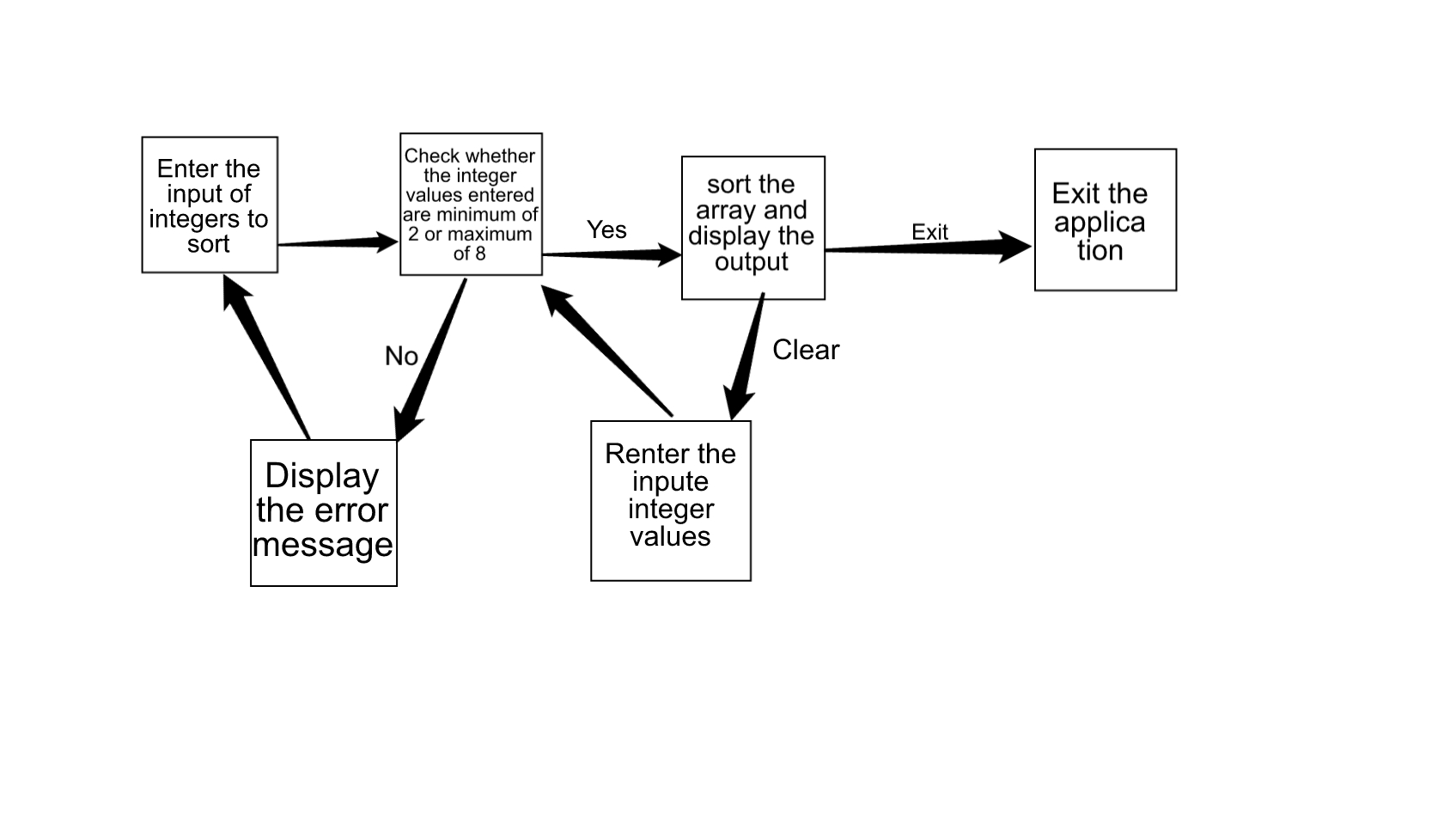
Architectural spike is an experiment which allows the users to know about the unknown elements in a user story. It is used to figure out answers for complicate technical and design process. When the user stories are provided in the product backlog that contains unknown elements which hinder the usable estimation and the developer finds it difficult then he should split the tasks into spikes to investigate the elements plus a user story to develop the functionality. The main goal of this architectural spike is reducing the risk of the technical problem or to increase the reliability of a user story’s estimates.

The main risk factor in the real time projects are the money and time. In our project the risk factor is the time as there is no cost is involved. So the cost in our project is the time as not finishing the project in time causes delay in the submission of the project which demands work under pressure.

**System metaphor**

System metaphor basically presents a logical architecture that makes use of some concept which is well known to the user and the developer to make them understand the system much better.

**System Metaphor Diagram**



**9.RELEASE PLANNING**

|  |  |
| --- | --- |
| **Pre-Iteration Learning**  30th August to 5th September | |
| **DATE** | **RELEASE** |
| 30TH | Install Android studio and learn programming in android |
| 31TH |
| 1ST |
|
|
| 2nd | Learn agile methodology scrum process |
| 3rd |
| 4th | Learn other agile method the extreme programming and its rules |
|
|
| 5th | We held a meeting to see what all have learned |
|
|
| Making of Product Backlog |
|
| Making design plans |
|
|

**Iteration planning**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ITERATION 1** | | **ITERATION 2** | | **ITERATION 3** | |
| **DATE** | **RELEASE** | **DATE** | **RELEASE** | **DATE** | **RELEASE** |
| **PLANNING** | | **PLANNING** | | **PLANNING** | |
| 6th | Sprint Backlog | 12TH | Sprint Backlog | 21ST | Sprint Backlog |
| Understanding Requirements Needs (i.e. HW1) | **MANAGING** | | **CODING** | |
| 19TH | Sprint Review | 21ST | Error Messages |
| 19TH | Sprint Retrospective | 22nd | Display Output |
| Creating User Stories | **DESIGNING** | | **TESTING** | |
|  | Requirements | 12TH | 3 Input values | 23rd | Unit Testing |
| **MANAGING** | | Separators/space |  | |
| 7th | Work-Products, Roles and Practices | 13TH | Input Range | 23rd | Sprint Review |
| 15TH | Checking for Sorting |  | Sprint Retrospective |
|  | Architecturally Influential Factors | **CODING** | | **DOCUMENTATION** | |
| 12TH | Inserting values into array | 24TH | Documenting the iteration |
| 8TH | Architectural Spike | Creating sorting array | 25TH | Compiling the whole document |
| 9th | Sprint Review | 14TH | Input Range between 0-9 | 26TH |
| Sprint Retrospective | 16TH | Check for sorting steps |  |  |
| **DESIGNING** | | **TESTING** | |  |
| 10th | Process Definition | 17TH | Unit testing |  |  |
| 18TH |
| 11th | Pre-Game | **DOCUMENTATION** | |
| 19TH | Documenting the iteration |
| 20TH |

**Release plan**

**10.TECHNOLOGY PREPARATION**

For a beginner, Android programming is more of java; so assuming that the developer is familiar with java, the developer would first need to understand some terminology for downloading and installing the software for the application development. Some of the basic terminology is as follows:

* **JDK:** The Java Development Kit is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.
* **API:** a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.
* **ADT:** Android Development Tools is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications.
* **SDK:** A software development kit (SDK) is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform.
* **IDE:** An integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion.
* **ANDROID STUDIO:** Android Studio is the official integrated development environment (IDE) for Android platform development. Based on Jet Brains' IntelliJ IDEA software, Android Studio is designed specifically for Android development.
* **EMULATOR**: In computing, an emulator is hardware or software that enables one computer system (called the host) to behave like another computer system (called the guest). An emulator typically enables the host system to run software or use peripheral devices designed for the guest system.

**Developing Phases:**

The steps for developing applications are as follows:

* **Setup:**

Install Android studio(<https://developer.android.com/studio/index.html>)

To install Android Studio on your Mac, proceed as follows:

* Launch the Android Studio DMG file
* Drag and drop Android Studio into the Applications folder, then launch Android Studio.
* Select whether you want to import previous Android Studio settings, then click **OK**.
* The Android Studio Setup Wizard guides you though the rest of the setup, which includes downloading Android SDK components that are required for development

**Development:**

**Writing your app:**

Android Studio includes tools for every stage of development, but what's most important is simply writing your app: writing the code, building layouts, creating images, and being productive along the way.

**Build and Run your app:**

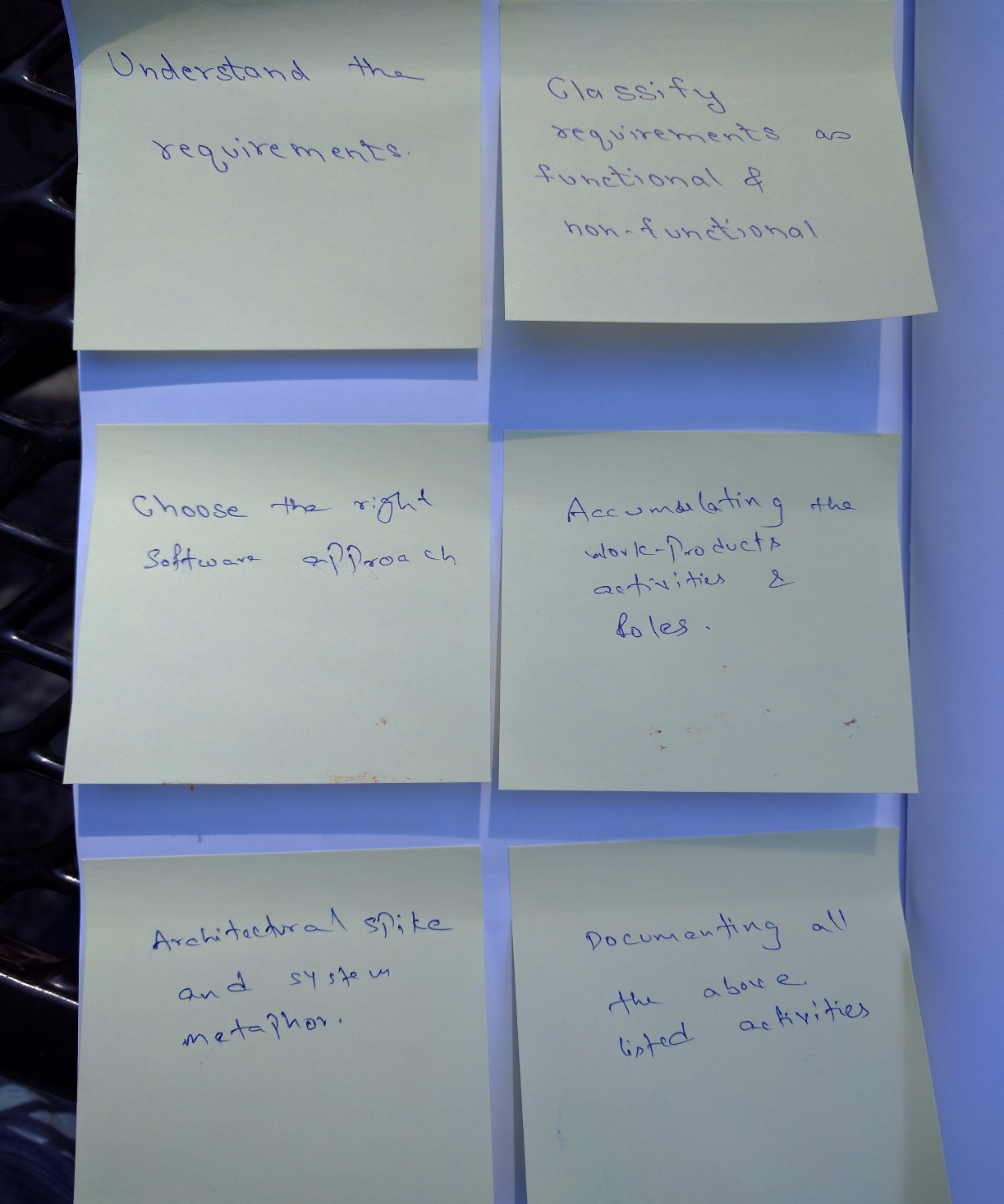
To build and run your app, click **Run**  https://developer.android.com/studio/images/buttons/toolbar-run.png. Android Studio builds your app with Gradle, asks you to select a deployment target (an emulator or a connected device), and then deploys your app to it. You can customize some of this default behaviour, such as selecting an automatic deployment target, by changing the run configuration.

If you want to use the Android Emulator to run your app, you need to have an Android Virtual Device (AVD) ready. If you haven't already created one, then after you click **Run**, click **Create New Emulator** in the **Select Deployment Target** dialog. Follow the Virtual Device Configuration wizard to define the type of device you want to emulate.

**11.DEVELOPMENT**

**ITERATION**

**User stories**

****

**Sprint planning:**

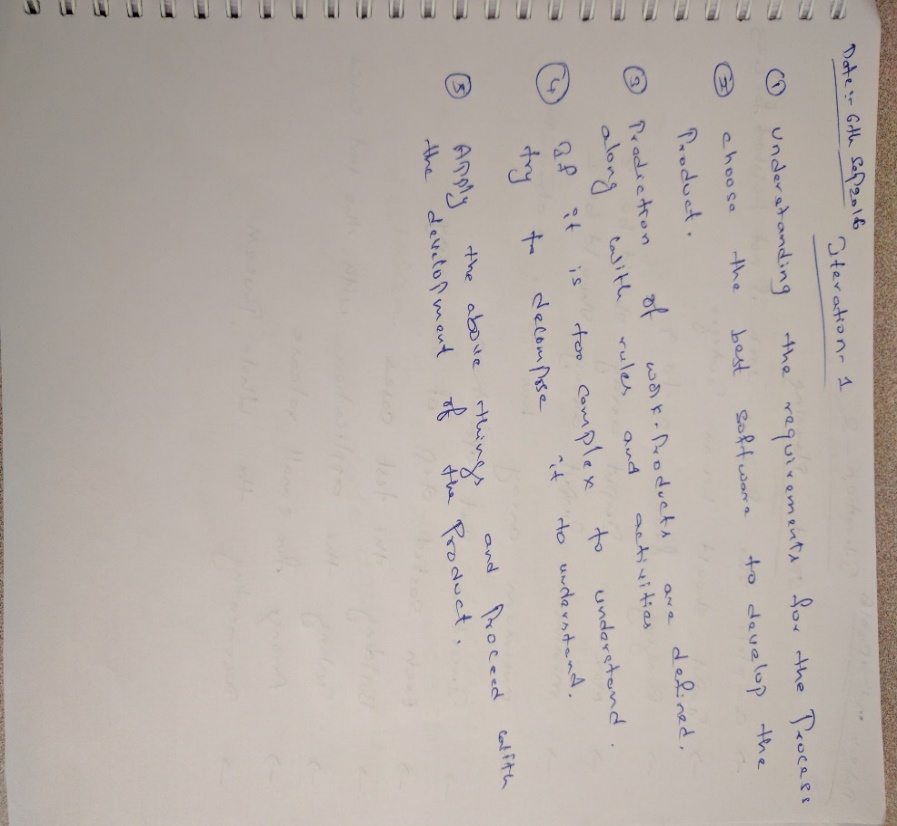
Before the initiation of each iteration (sprint), two consecutive meetings are held.

In the first, stakeholders (includes scrum master, entire agile team, and customer) meet to refine and re-prioritize the product backlog and release backlog, and to choose goals for the next iteration, usually driven by highest business value and risk.

In the second meeting, the scrum team and customer meet to consider how to achieve the requests, and to create a sprint backlog of tasks to meet the goals. If estimated effort exceeds resources, another planning cycle occurs.

**Sprint planning meeting:**

For the first sprint planning meeting, we agreed to meet in the library for the fixed duration of 2 hours. We did write down the list of to-do things for the 1st sprint iteration. Below is the highlights of our first sprint planning meeting.



|  |  |  |  |
| --- | --- | --- | --- |
| Task description | Originator | Responsible | Estimated Time  (hours) |
| Understand the requirements clearly | Ashish | Ashish, Harsha, Shivam, Yamini | 4 |
| Create user story cards | Shivam | Shivam, Harsha, Yamini | 2 |
| Classify requirements as functional and nonfunctional requirements | Harsha | Harsha | 1 |
| Document the requirements | Ashish | Ashish, Harsha | 3 |
| Choose the right software process approach | Yamini | Yamini, Shivam, Harsha, Ashish | 2 |
| Accumulate the work-products, activities and roles | Harsha | Harsha, Shivam | 2 |
| Architecture spikes and system metaphor | Yamini | Yamini, Shivam, Ashish | 2 |
| Document all the above listed activities | Shivam | Shivam, Yamini, Harsha, Ashish | 3 |

**Fig : Sprint Backlog Iteration**

**Managing:**

The first iteration is all about pre-game activities. Looking at the agenda, we felt that there is more to learn before we can actually start developing the product. But, the flexibility in agile eased our work.

In the first iteration, we did not focus on product development, however it took us more time than we expected for understanding the requirements clearly.

To keep track of the progress, we agreed to meet once in 3 days though we stayed connected via hangout and skype. The regular meetings would be in the afternoon, from 12.30 pm to 1.15 pm, in Pollak library.

**Designing:**

In design phase, we tried to decompose the tasks using system metaphor, which is the concept of architectural spike. Architectural spike taught us how to make our product more reliable by making us reduce technical abnormalities.

Understanding Requirements

User Stories

Process Definition

Workproduct, roles, activities

architecturally influential factors

pre-game

document the requirements

non-functional requirements

functional requirements

architectural spike

**Coding:**

As discussed earlier, we are not focusing on coding in this iteration. As there is no coding, the Testing and the Small Release will be showcased in the forthcoming iterations.

**Daily scrum meetings:**

The daily scrum meetings would be via Skype or Google hangout. The scrum meeting is usually done in the morning from 11.00 am to 11.15am. This meeting is very short because, this meeting is used to discuss about the day’s work, and at rare times, about previous day’s work.

**Sprint review:**

Sprint review is a meeting hosted by scrum master, that is held to review the scrum activities for that iteration. All the stake holders must be present during this meeting.

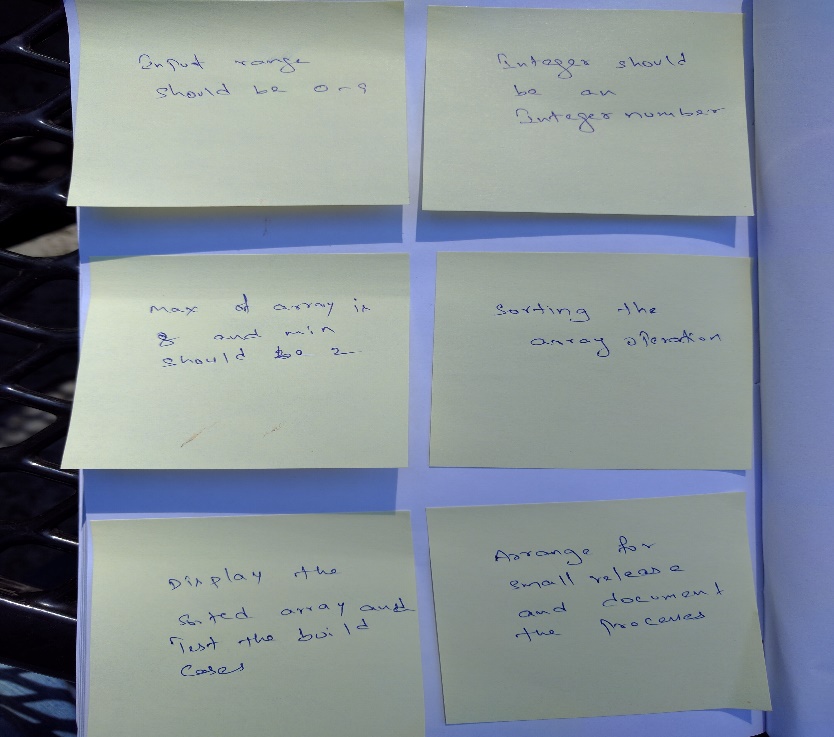
Well in sprint review, the scrum team shows what they have accomplished during the sprint, on basis of sprint backlog. Also if applicable, the scrum team can also show their accomplishments based on product backlog.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task description | Originator | Responsible | status | Actual Time taken  (hours) |
| Understand the requirements clearly | Ashish | Ashish, Harsha, Shivam, Yamini | completed | 5 |
| Create user story cards | Shivam | Shivam, Harsha, Yamini | completed | 1.5 |
| Classify requirements as functional and nonfunctional requirements | Harsha | Harsha | completed | 1 |
| Document the requirements | Ashish | Ashish, Harsha | completed | 2 |
| Choose the right software process approach | Yamini | Yamini, Shivam, Harsha, Ashish | completed | 1 |
| Accumulate the work-products, activities and roles | Harsha | Harsha, Shivam | completed | 2 |
| Architecture spikes and system metaphor | Yamini | Yamini, Shivam, Ashish | Completed | 2 |
| Document all the above listed activities | Shivam | Shivam, Yamini, Harsha, Ashish | In progress | 3 hours and counting |

**Revised sprint backlog**

**Iteration 2**

**User stories**

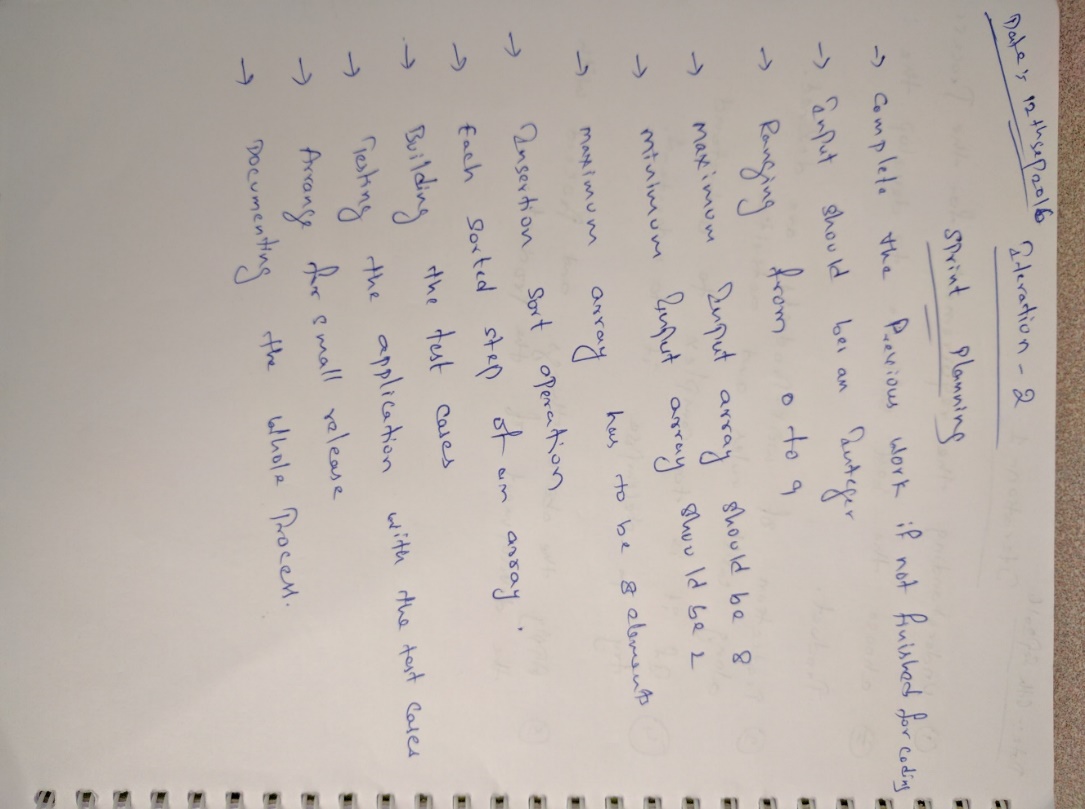
****

**Sprint planning:**

In the previous iteration, we were unable to complete the activities within the given time such as design phase. So, the unfinished work will be taken up in the second iteration as the first priority. Along with this, we would start the main task i.e development of the application.

**Sprint planning meeting:**

For the second sprint planning meeting, we agreed to meet at Harsha’s place for the fixed duration of 3 hours. We did write down the list of to-do things for the 2nd sprint iteration. Below are the highlights of our second sprint planning meeting.



**Sprint backlog:**

|  |  |  |  |
| --- | --- | --- | --- |
| Task description | Originator | Responsible | Estimated time  (hours) |
| Input range [0 to 9] | Yamini | Shivam, Yamini | 1 |
| Input is an integer number | Ashish | Harsha, Ashish | 0.5 |
| Minimum input array size is 2 | Shivam | Shivam, Harsha | 0.5 |
| Maximum input array size is 8 | Shivam | Shivam, Harsha | 0.5 |
| Sort operation | Yamini | Yamini, Ashish | 1 |
| Display array contents | Harsha | Harsha, Yamini | 1 |
| Build test cases | Harsha | Harsha, Shivam, Ashish, Yamini | 2 |
| Test the developed product | Ashish | Yamini, Shivam, Harsha, Ashish | 2 |
| Arrange for the small release | Yamini | Yamini | 0.5 |
| Document the process | Ashish | Yamini, Shivam, Harsha, Ashish | 3 |

**Fig: Sprint Backlog Itertion-2**

**Managing:**

After gathering the requirements and user stories, it is time to jump to the coding. For the first release, we decided to build and release the beta application of Sorting application. This beta application includes just the basic operation like, collecting the input, process it, and display the sorted integers.

**Designing:**

Enter 2 or 8 elements

Spaces between integers

Input range 1-9

Sorting the entered integers

Unit testing

Documentation

**Coding:**

**package** com.example insersort;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }  
  
 **public void** onSortClick(View v)  
 {  
 **if**(v.getId() == R.id.***Bsort***)  
 {  
  
 EditText inputNum = (EditText)findViewById(R.id.***TFnum***);  
  
  
 String inputNumStr = inputNum.getText().toString();  
  
 String []inputNumStrs = inputNumStr.split(**" "**);  
  
 **int** count = inputNumStrs.**length**;  
  
 *//if(count < 2 || count > 8 ) {  
 //Toast.makeText(getApplicationContext(),"Input number count should be greater than 1 and less than 9",Toast.LENGTH\_SHORT).show();  
 //} else {  
 //String inputNumStr = inputNum.getText().toString();  
 //EditText n = (EditText)findViewById(R.id.TFnumb);  
 //int n = Integer.parseInt(String.valueOf(count));  
 //String inputNumStr = inputNum.getText().toString();* **int** []inputArry = **new int**[count];  
  
 *//String []inputNumStrs = inputNumStr.split(" ");  
 //System.out.println("hey");  
 //System.out.println(inputNumStrs);* String outputStr = **""**;  
  
 **if**(inputNumStrs.**length** == count){  
 **for**(**int** i = 0;i<count;i++){  
 inputArry[i] = Integer.*parseInt*(inputNumStrs[i]);  
 *// Integer N = Integer.parseInt(inputNumStrs[i]);  
 //System.out.println("hey");  
 //System.out.println(N);* }  
  
 *//if(chkInputVal(inputArry,count)){* **for**(**int** j=1;j<=count-1;j++){  
 **int** x = inputArry[j];  
 **int** k = j - 1;  
 **while**(k >= 0 && inputArry[k] > x){  
 inputArry[k+1] = inputArry[k];  
 k = k - 1;  
 }  
 *//for(int j=1;)* inputArry[k+1] = x;  
 outputStr = outputStr.concat(inputToStr(inputArry,count));  
 }  
  
 *//String outputStr = "";  
 //for (int y = 0; y<count;y++){  
 //outputStr = outputStr.concat(String.valueOf(inputArry[y]));  
 //outputStr = outputStr.concat(" ");  
 //}* EditText res = (EditText)findViewById(R.id.***TFres***);  
 res.setText(outputStr);  
 *// }  
 //else {  
 //Toast.makeText(getApplicationContext(),"Input numbers greater than 9",Toast.LENGTH\_SHORT).show();  
 //}* } **else** {  
 Toast.*makeText*(getApplicationContext(),**"Input numbers not matching N"**,Toast.***LENGTH\_SHORT***).show();  
 }  
  
  
 *//}* }  
 }  
  
 **public void** onClearClick(View v){  
 *// EditText n = (EditText)findViewById(R.id.TFnumb);* EditText inputNum = (EditText)findViewById(R.id.***TFnum***);  
 EditText res = (EditText)findViewById(R.id.***TFres***);  
 *// n.setText("");* inputNum.setText(**""**);  
 res.setText(**""**);  
 }  
  
 **public void** onExitClick(View v){  
 Intent intent = **new** Intent(Intent.***ACTION\_MAIN***);  
 intent.addCategory(Intent.***CATEGORY\_HOME***);  
 intent.setFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK***);  
 startActivity(intent);  
 }  
  
 **public** String inputToStr(**int** []inputArry, **int** count){  
 String outputStr = **""**;  
 **for** (**int** y = 0; y<count;y++){  
 outputStr = outputStr.concat(String.*valueOf*(inputArry[y]));  
 outputStr = outputStr.concat(**" "**);  
 }  
 outputStr = outputStr.concat(**"\n"**);  
 **return** outputStr;  
 }  
  
   
 }  
  
}

**Testing:**

“Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application product”.

Whereas “a test case, in software engineering, is a set of conditions under which a tester will determine whether an application, software system or one of its features is working as it was originally established for it to do”.

The following are the test cases, and their corresponding result of our beta product.

Test case 1: Inputs must be in the range of 0 to 9. Any other inputs must be ignored.

Status: completed Result: passed

Test case 2: Input must be an integer value

Status: completed Result: passed

Test case 3: The application should not accept more than 8 integers.

Status: completed Result: passed

Test case 4: The application should not accept less than 2 integers.

Status: completed Result: passed

Test case 5: The array size must be 8.

Status: completed Result: passed

Test case 5: Display the sorted steps of the array

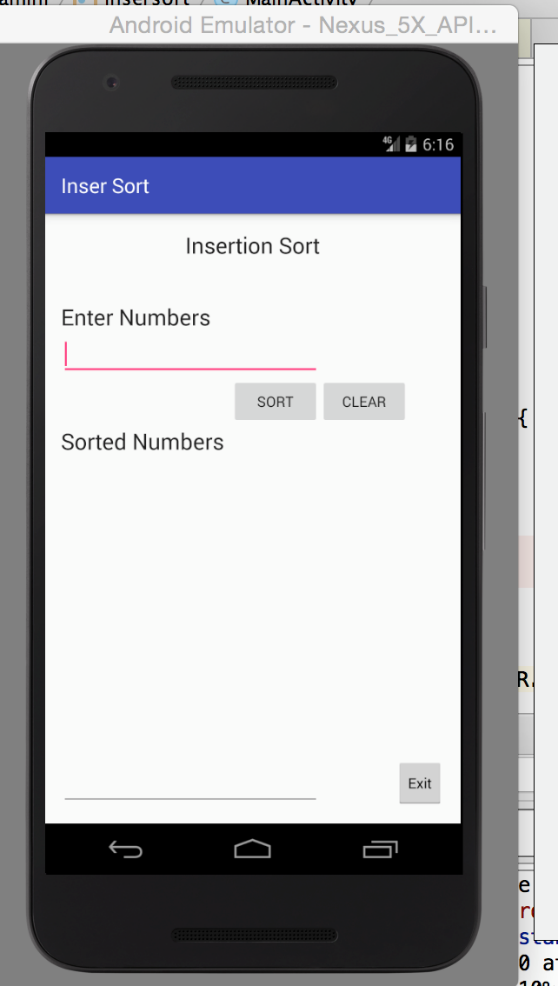
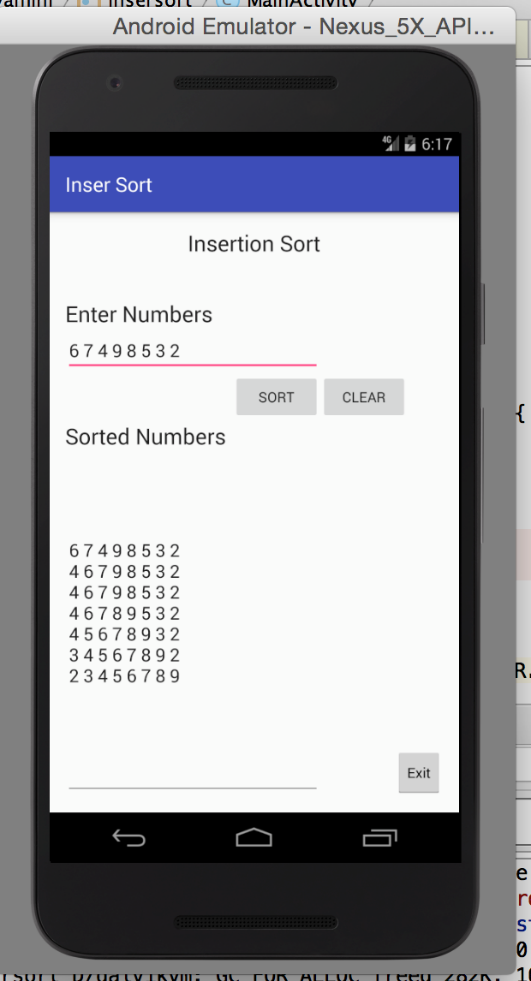
Status: completed Result: passed

Test case 6: Display the sorted array

Status: completed Result: passed

**Small release:**

For the beta release, we did enact the process of beta product release, in the exact same way the software developers do. Harsha being the user, Ashish being the scrum manager, Yamini being the developer and Shivam was the tester. It was indeed the best experience for all of us.

**Daily scrum meetings:**

The daily scrum meetings would be via Skype. For this iteration, we slightly changed the meeting time. The scrum meeting is held daily in the morning from 10.00 am to 10.15am, instead of 11.00 am to 11.15 am.

**Sprint review:**

The second iteration lasted for 9 days, as it involved coding and testing. Also, it costed us more time because, we did enact the process of beta product release in the exact same way the software developers do.

Below is the revised version of Sprint backlog that we built in the beginning of this iteration. All activities in this iteration is successfully accomplished, according to our estimated plan.

**Sprint backlog:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task description | Originator | Responsible | Status | Time taken  (hours) |
| Input range[0 to 9] | Yamini | Shivam, Yamini | Completed | 1 |
| Input is an integer number | Ashish | Harsha, Ashish | Completed | 0.5 |
| Minimum input array size is 2 | Shivam | Shivam, Harsha | Completed | 0.5 |
| Maximum input array size is 8 | Shivam | Shivam, Harsha | Completed | 0.5 |
| Sort operation | Yamini | Yamini, Ashish | Completed | 1 |
| Display array contents | Harsha | Harsha, Yamini | Completed | 1 |
| Build test cases | Harsha | Harsha, Shivam, Ashish, Yamini | Completed | 2 |
| Test the developed product | Ashish | Yamini, Shivam, Harsha, Ashish | Completed | 2 |
| Arrange for the small release | Yamini | Yamini | Completed | 0.5 |
| Document the process | Ashish | Yamini, Shivam, Harsha, Ashish | Completed | 3 |

**Product backlog refinement:**

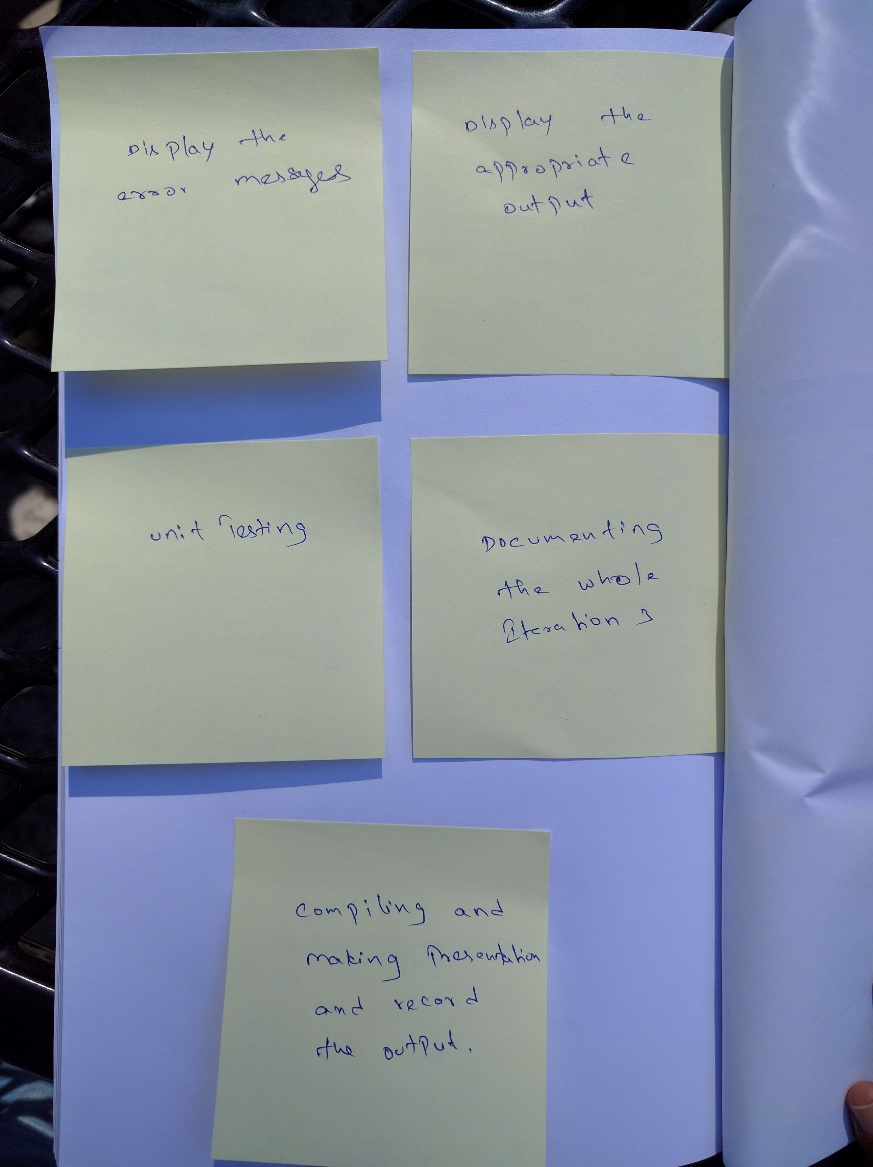
Most of the scrum activities is exactly similar to the activities present in product backlog. So, the product backlog can be refined by eliminating the activities which are successfully built using scrum.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requirement | Number | category | status | priority | Estimate  (hours) |
| Input is an integer number | 1 | Requirement | Completed | High | 1 |
| Input range[0 to 9] | 2 | Constraint | Completed | High | 1 |
| Minimum array size is 2 | 3 | Constraint | Completed | Medium | 0.5 |
| Maximum array size is 8 | 4 | Constraint | Completed | Medium | 0.5 |
| Sort | 9 | Requirement | Completed | high | 1 |
| Show error message | 5 | Enhancement | Completed | High | 2 |
| Display sorted array | 6 | Requirement | Completed | Medium | 1 |
| User friendly GUI | 7 | Requirement | Completed | High | 2 |
| Must be an android application | 8 | Requirement | Completed | High | 7 |

**Refined version of product backlog**

**Iteration 3**

**User stories**

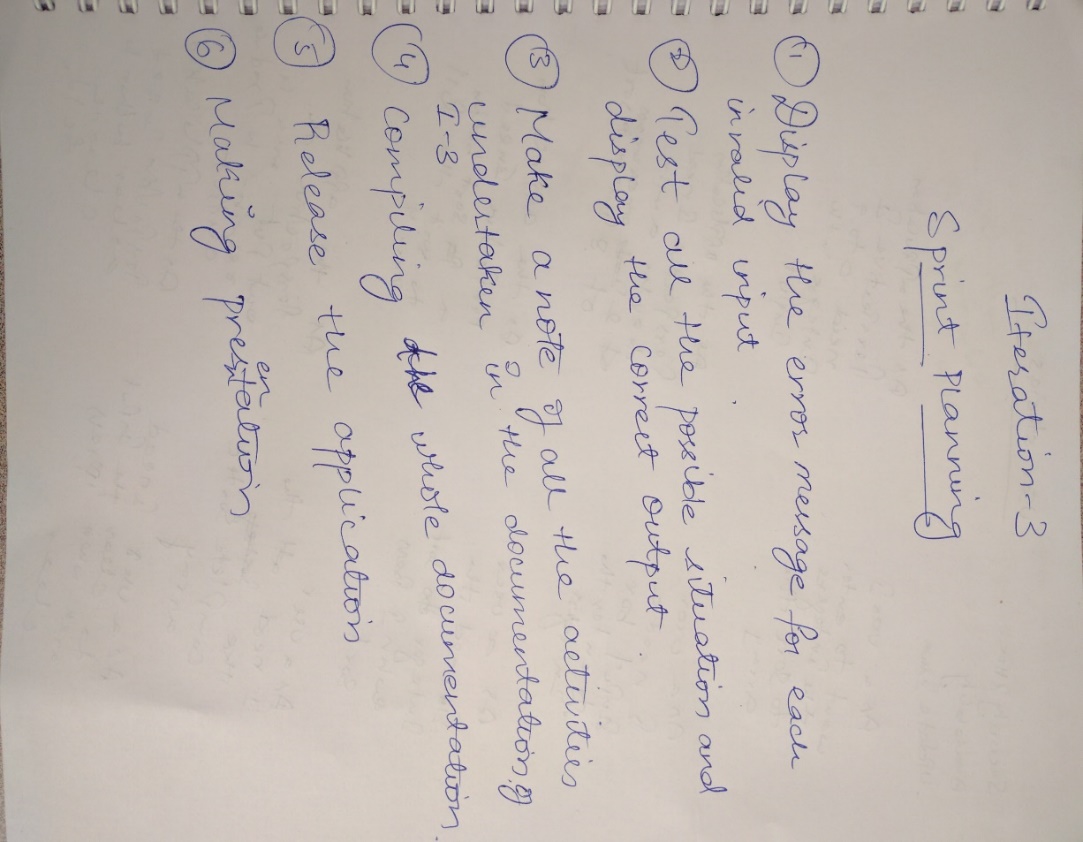
****

**Sprint Planning**

Each sprint starts with a timely meeting called sprint planning. In this meeting, the Scrum team selects and understands the work to be done in the sprint. The recommended time for the sprint planning meeting is maximum of two hours.

**Sprint Planning Meeting**

* We had a meeting for a fixed amount of time and scrum team selected the work to be done by them in the sprint.
* Along with the product backlog, we developed a sprint backlog for the sprint.
* This is the last iteration and we estimated to finish coding, testing and compile reports in this iteration.



**Sprint Planning Meeting-iteration3**

**Sprint Backlog**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User story** | **Number** | **Category** | **Status** | **Priority** | **Estimates  (in hours)** |
| Display error messages | 1 | Enhancement | Incomplete | Medium | 3 |
| Display appropriate output | 2 | Requirement | Incomplete | High | 5 |
| Unit testing | 3 | Testing | Incomplete | High | 2.5 |
| Documenting this iteration | 4 | Documentation | Incomplete | Medium | 2 |
| Compiling the whole document | 5 | Documentation | Incomplete | Medium | 6 |
| Making presentation and recording it | 6 | Presentation | Incomplete | High | 5 |

**Sprint Backlog-iteration3**

**Designing**

Enter the input inergers

Check all the conditions

Display error messages

Documenting the project

**Coding**

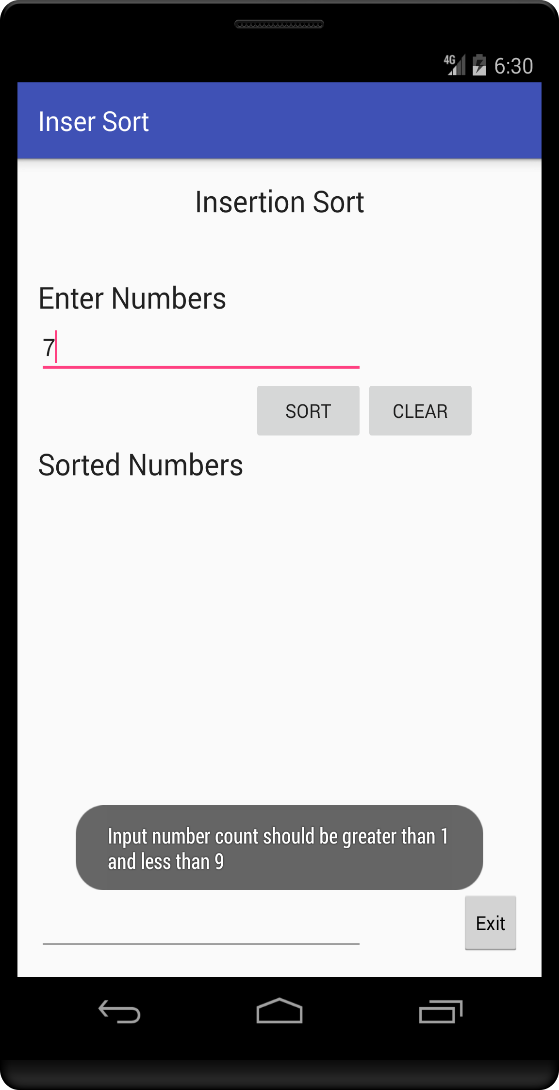
**package** com.example.yamini.insersort;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }  
  
 **public void** onSortClick(View v)  
 {  
 **if**(v.getId() == R.id.***Bsort***)  
 {  
  
 EditText inputNum = (EditText)findViewById(R.id.***TFnum***);  
  
  
 String inputNumStr = inputNum.getText().toString();  
  
 String []inputNumStrs = inputNumStr.split(**" "**);  
  
 **int** count = inputNumStrs.**length**;  
  
 **if**(count < 2 || count > 8 ) {  
 Toast.*makeText*(getApplicationContext(),**"Input number count should be greater than 1 and less than 9"**,Toast.***LENGTH\_SHORT***).show();  
 } **else** {  
 *//String inputNumStr = inputNum.getText().toString();  
 //EditText n = (EditText)findViewById(R.id.TFnumb);  
 //int n = Integer.parseInt(String.valueOf(count));  
 //String inputNumStr = inputNum.getText().toString();* **int** []inputArry = **new int**[count];  
  
 *//String []inputNumStrs = inputNumStr.split(" ");  
 //System.out.println("hey");  
 //System.out.println(inputNumStrs);* String outputStr = **""**;  
  
 **if**(inputNumStrs.**length** == count){  
 **for**(**int** i = 0;i<count;i++){  
 inputArry[i] = Integer.*parseInt*(inputNumStrs[i]);  
 *// Integer N = Integer.parseInt(inputNumStrs[i]);  
 //System.out.println("hey");  
 //System.out.println(N);* }  
  
 **if**(chkInputVal(inputArry,count)){  
 **for**(**int** j=1;j<=count-1;j++){  
 **int** x = inputArry[j];  
 **int** k = j - 1;  
 **while**(k >= 0 && inputArry[k] > x){  
 inputArry[k+1] = inputArry[k];  
 k = k - 1;  
 }  
 *//for(int j=1;)* inputArry[k+1] = x;  
 outputStr = outputStr.concat(inputToStr(inputArry,count));  
 }  
  
 *//String outputStr = "";  
 //for (int y = 0; y<count;y++){  
 //outputStr = outputStr.concat(String.valueOf(inputArry[y]));  
 //outputStr = outputStr.concat(" ");  
 //}* EditText res = (EditText)findViewById(R.id.***TFres***);  
 res.setText(outputStr);  
 }  
 **else** {  
 Toast.*makeText*(getApplicationContext(),**"Input numbers greater than 9"**,Toast.***LENGTH\_SHORT***).show();  
 }  
  
  
  
 } **else** {  
 Toast.*makeText*(getApplicationContext(),**"Input numbers not matching N"**,Toast.***LENGTH\_SHORT***).show();  
 }  
  
  
 }  
  
 }  
 }  
  
 **public void** onClearClick(View v){  
 *// EditText n = (EditText)findViewById(R.id.TFnumb);* EditText inputNum = (EditText)findViewById(R.id.***TFnum***);  
 EditText res = (EditText)findViewById(R.id.***TFres***);  
 *// n.setText("");* inputNum.setText(**""**);  
 res.setText(**""**);  
 }  
  
 **public void** onExitClick(View v){  
 Intent intent = **new** Intent(Intent.***ACTION\_MAIN***);  
 intent.addCategory(Intent.***CATEGORY\_HOME***);  
 intent.setFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK***);  
 startActivity(intent);  
 }  
  
 **public** String inputToStr(**int** []inputArry, **int** count){  
 String outputStr = **""**;  
 **for** (**int** y = 0; y<count;y++){  
 outputStr = outputStr.concat(String.*valueOf*(inputArry[y]));  
 outputStr = outputStr.concat(**" "**);  
 }  
 outputStr = outputStr.concat(**"\n"**);  
 **return** outputStr;  
 }  
  
 **public boolean** chkInputVal(**int** []inputArry, **int** count){  
 **boolean** flag = **true**;  
 **for**(**int** i = 0;i<count;i++){  
 **int** x = inputArry[i];  
 **if**(x < 0 || x > 9){  
 flag = **false**;  
 }  
  
 }  
 **return** flag;  
 }  
  
}

**Testing**

In iteration 3, the application passed and is displaying the proper result as below:

* Meets the requirements that guided its design and development.
* Responds correctly to all kinds of inputs.
* Performs its functions within an acceptable time.
* It is sufficiently usable.
* Can be installed and run in its intended [environments](http://en.wikipedia.org/wiki/Operating_environment).
* Achieves the general result, which is stakeholder’s desire.

**Small Release**

****

**Sprint Review**

In this iteration, we completed the coding of the application. Since we already did 85% of the coding in previous iteration, it was actually easier to finish the remaining part of it.

**Product Increment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User story** | **Number** | **Category** | **Status** | **Priority** | **Estimates  (in hours)** |
| Display error messages | 1 | Enhancement | Complete | Medium | 3 |
| Display appropriate output | 2 | Requirement | Complete | High | 5 |
| Unit testing | 3 | Testing | Complete | High | 2.5 |
| Documenting this iteration | 4 | Documentation | Complete | Medium | 2 |
| Compiling the whole document | 5 | Documentation | Complete | Medium | 8.5 |
| Making presentation and recording it | 6 | Presentation | Complete | High | 2.5 |

**Product increment-iteration3**

**Sprint Retrospective**

Overall, iteration 3 was easy to finish without any difficulty. We scheduled the development in such a way that our team don’t have to rush to the last iteration and we were successful in accomplishing what we planned.

**12. ACCEPTANCE TESTING**

Acceptance testing is a test that checks whether the software system has met the requirement specifications or not. In our project we had 3 iterations in which we test the application at the end of each iteration just to make sure that it works as per the specified the requirement. Following are test cases that tests the functionality of user stories.

**Test Case 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Input value | Expected | Actual | Status |
| The user entered less than 2 input values, then entered the sort key. | 7 | The application detects the error and displays a toast error message “The input should be more than 1 and less than 9” | All of the expected results were met. | Pass |

**Test Case 2**

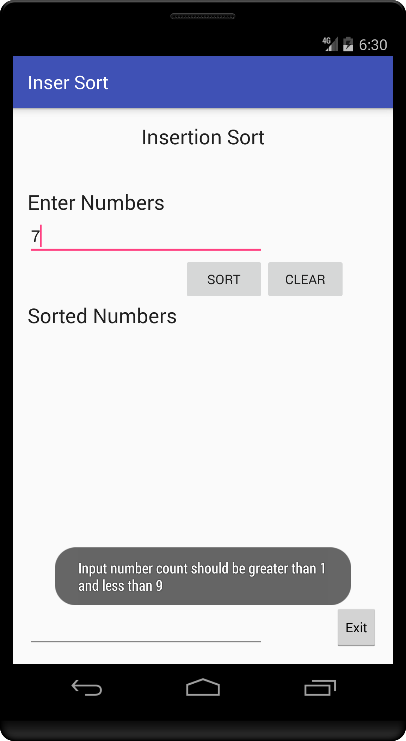
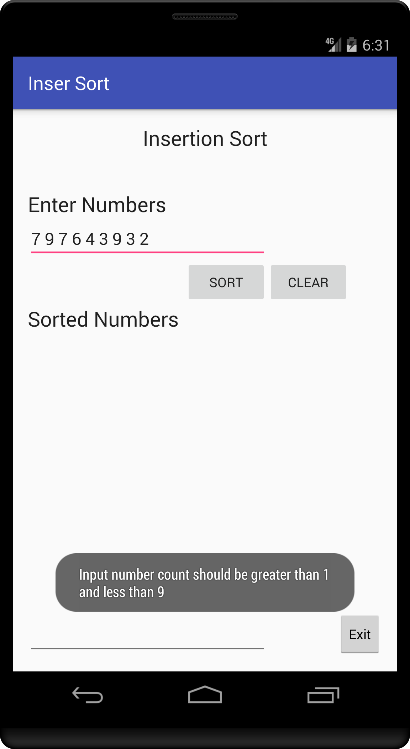
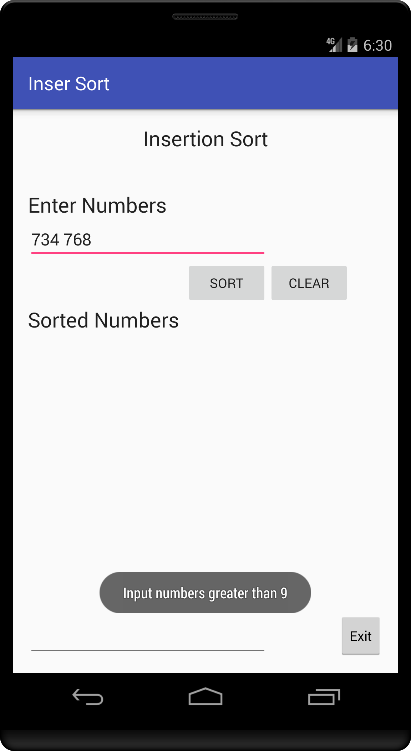
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Input value | Expected | Actual | Status |
| The user entered more than 8 input values, then entered the sort key. | 7 9 7 6 4 3 9 3 2 | The application detects the error and displays a toast The input should be more than 1 and less than 9” | All of the expected results were met. | Pass |

**Test Case 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Input value | Expected result | Actual result | Status |
| The user entered a two-digit number that it is greater than 9, then entered the sort key. | 22 | The application detects the error and displays a toast error message “the number should be greater than 1 and less than 9”. | All of the expected results were met. | Pass |

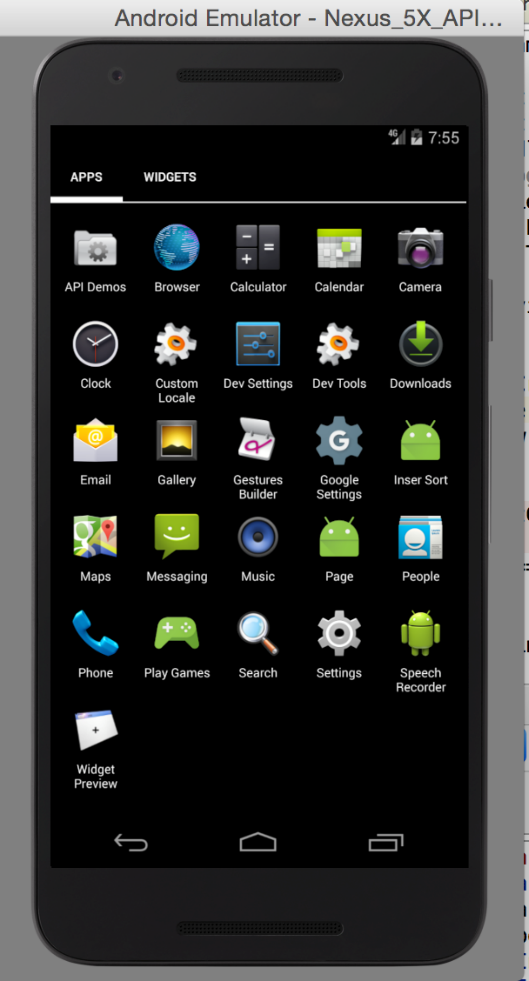
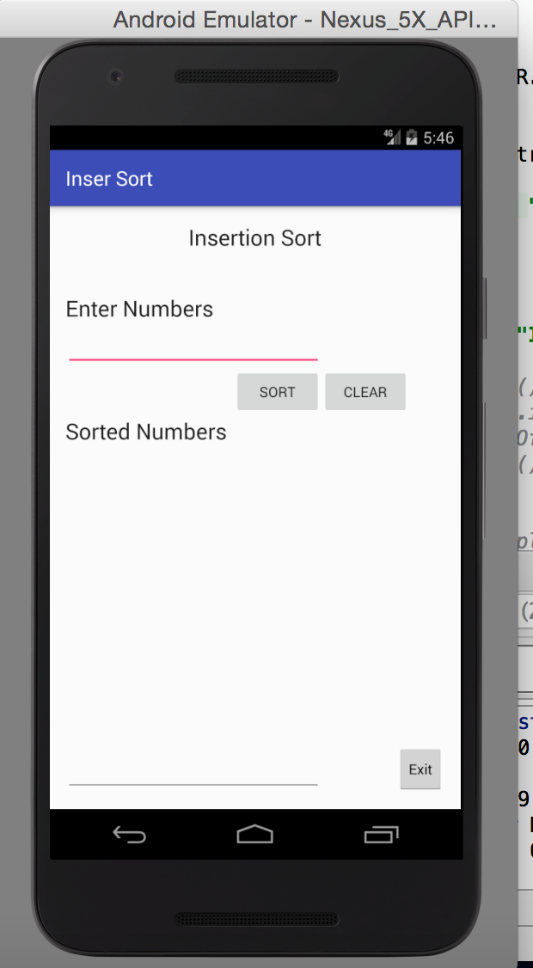
**Test Case 4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Input value | Expected result | Actual result | Status |
| The user entered integers array greater than 9, then entered the sort key. | 734 768 | The application detects the error and displays a toast error message “the numbers greater than 9.” | All of the expected results were met. | Pass |

**13.RELEASE**

The Insertion Sorting application final release.

** **

**14.OTHER ACTIVITIES**

1. **Estimation**

In our project the estimated activity is integrated throughout the development cycle of our application. We estimated the time for every type of input that a user can provide and the respective output that the application can be produced by the application for the project. An estimation procedure has been designed to best fit our resources (time, development).

To adjust the project scope to available budget and resources is important. Depending up on the resources and time availability we balance the estimations and thus relying on the platform of the development and design of our code we adjusted the estimates of the inputs and the relevant outputs of the application and the behavior of the application and time required for the completion of the project.

Practical project planning helped us to create improved estimates and create critical path for project implementation and scheduling in which we can expect the time for the completion. The proper completion of our project iterations were possible through estimates and this helped us to create a breakdown structure between team members in our project.

1. **Project Planning**

When a project plan is documented properly, it can be used as a standard benchmark to compare with the actual performance by following the planning principles

* We understood the requirements properly by interacting with end users. The user wanted to sort the inserted array of elements through the insertion sorting.
* At each subsequent requirement refinement, resource projections, size estimates, and schedules are also refined as requirements went on changing by planning different type of functionalities to the application for example adding functionality of output.
* When the requirement became clear we planned the detailed design of the project and incorporated the same in our project.
* As various parts of the project become sufficiently well understood, implementation and details are established and documented in further plan refinements.
* Throughout this cycle, the plan provides the framework for negotiating the time and resources to do the job.

**3.Project Monitoring and Control**

In the project this is the process which monitors and controls the process that is used to initiate, plan execute, and close the project. In this we followed three steps to process and monitor the application for efficient results on sorting the array of integers.

* **Scope verification control**

This is defined by the manager to verify and control the elements in the project which decreases or increases the scope. When we need to display more information in the output which might be beneficial for the project we need to seek the approval from the stake holders or the stake holders might not agree and that would be the same discarding some information from the output.

* **Schedule Control**

In the project once the schedule has been determined it has to be followed. If there are any influential factors that might ensure the changes are beneficial then the scheduler is required for approval as adding more functionalities means adding more time to complete the project. So the scheduler determines whether the changes are beneficial or not then accordingly he grants the permission.

* **Cost Control**

Describing the factors which influences the cost changes that ensures contributing more time and money for adding more functionalities like adding more buttons in the sorting application.

* **Quality Control**

The monitoring of the results is to be continuous and comparing them with standard results to maintain the quality of the software. We determine the outputs in the iterations after the inputs are defined then we compare them with the standard output.

**4.Configuration Management**

* We maintained baseline and followed the same eclipse and android SDK version among all the developers to overcome the configuration conflict after each set of enhancement and each baseline is maintained.
* We Tested code developed by each developer and approved changes in the baseline helps managing the configuration
* We maintained single official copy of code which helped managing their configuration and have been tracking of each revision which helped to resolve problems occurred at each new level, the problem was solved by returning to previous tests
* Keeping track of every change to every module and test case helps to manage the configurations and resolve all the problems regarding configurations.

**5. Quality Assurance**

Quality assurance (QA) consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control activities. We have to deploy work of software quality assurance team and the team must ensure the following steps are ensured

* The project team conducted independent reviews for the performance of the application on determining the sorting when given the integer arrays as input.
* We documented each set of input and output for purpose of maintenance and enhancement.
* We checked each mechanism are in place and each function is working as planned.

Our development team made sure each software task is successfully completed before the succeeding one is begun.

* Ensure that all required tests are performed and test reports produced.
* Participate in all project quarterly and phase reviews and register non-concurrence if the appropriate standards and procedures have not been reasonably followed.

**6. Risk management**

In this activity we identified, addressed and tried to eliminate software risks or threats to successful software operation of the application. The loss is often considered in terms of direct financial loss, but also can be a loss in terms of credibility, future business, and loss of property or life. The risk management process can be broken down into two interrelated phases, risk assessment and risk control

* In first phase we have to identify different types of risks analyze them and prioritize them during the development period. we considered various factors like People, size, process, technology, tools, organizational, managerial, customer, estimation, sales, support.

1. **Risk identification**

In this step considering all the factors, we identified and divided the types of risks like

1. **Generic risks**

According to our project’s view we determined the generic risk as losing a key person in the development team who knows the design of overall application and can cause severe damage or failure of the project as the person who will be developing the main function cannot leave the project.

1. **Product-specific risks**

We studied the present market for the similar kind of application or product which can cause business risk that threaten the viability of the software, such as building an excellent product no one wants or building a product that no longer fits into the overall business strategy of the company.

* Second Phase

1. **Analyze**

After risks have been identified and enumerated, the next step is risk analysis. Through Risk analysis, we transform the risks that were identified into decision-making Information. In turn, each risk is considered and a judgment made about the probability and the seriousness of the risk.

1. **Prioritize**

We sorted the list so that the high probability, high impact risks percolate to the top of the table and the low-probability, low impact risks drop to the bottom.

1. **Plan**

After prioritizing the risks each risk, information is gathered by investigation and then action is chosen to overcome each risk. Planning on the analyzed risks reduces the risks and finally there will be stage called rick acceptance where the team accepts the risk and incurred loss.

The following are some risks that we identified in our project:

|  |  |
| --- | --- |
| **Risk** | **Probability** |
| A member has insufficient knowledge about android programming language. | Medium |
| Some group members may not contribute evenly to the group. | Low |
| Member may fall ill | Low |
| Miss the project deadline. | Low |
| The application may crash | Low |

**Risks**

**15. LESSONS LEARNED**

From our experience in this home work we were able to use extreme programming (XP) and scrum processes together by apply them on the Insertion Sorting application. Also, we found it interesting to learn about Android programming. When we started the home work we faced some difficulties on defining our own process. However, by studying and discussing the two processes and make sure that each of us know exactly what each process is, we started to create our own process. After that, as a team we defined our roles and responsibilities to successfully build the product and document it. Regardless of the lack of resources to practice scrum and XP processes together, we gained a lot of experience on how to practice them in real. The following is the experience that each member gain:

**Sree Harsha**

While doing this homework I gained a lot of knowledge and experience about agile process XP and scrum processes. By learning them and applying them in the process to complete our homework I got to learn many things about how to apply XP and scrum process together in one project. The other thing is I had no experience in android programming before this this assignment helped to get to introduce to the android platform and gain good amount of knowledge in android programming and improve my skillset. To complete this project, it took a great amount of team coordination and the team members were on right track without much delays.

**Shivam Rana**

During this homework, I got more knowledge about the Android Developing. Earlier also I have developed android apps using Eclipse but this time, I experienced something new i.e. Android Studio which is better than Eclipse. Before starting this homework, I was not having any idea about XP and Scrum. But as time passed, I got to know about XP and Scrum and I used it in completing this homework. Also as I have not worked in an IT Company before so I was not having any idea about how to build a software from scratch. But this project helped me in gaining the knowledge’s about those standards. My team members were very supportive and since my team members including me have a clear understanding it was easy to complete this project.

**Ashish Paul**

Due to this homework, I was able to understand how a software process initiated. Though I’m a CS undergrad, I was not taught how the main concepts such as Scrum and XP even fitted in the software development. Though I understood a lot about them by doing a lot of research but it was quite clear when I kept this practice to use. Developing and documenting the project gave me clear insights on how the whole process was being developed from the scratch to a full version. Adding up to my experience, I had one of the best times learning Android which was quite similar to Java that I had worked on before which made this whole adventure even more better. My team was one of the best people I had ever worked with as it clearly showed how to be a team’s man and how to complete a project in real time.

**Yamini**

Before starting the project, I had a basic knowledge about Scrum and Extreme programming but after we started with the project I could use the basic knowledge that i gained on a practical level. As a developer, I learnt to build an app using android studio and gained a lot of experience. Also, it helped me improve my programming skills. This was my first time working with a project team. It required some amount of coordination. With the help of my team mates and proper understanding it was easy to complete the project

**16. REFERENCES**

* Humphrey, Watts S., Managing the Software Process
* https://developer.android.com/studio/index.html
* http://techterms.com/definition/android
* https://en.wikipedia.org/wiki/Systemcontextdiagram
* <https://en.wikipedia.org/wiki/Project_stakeholder>
* <https://en.wikipedia.org/wiki/Testcase>
* http://www.selectbs.com/process-maturity/what-is-extreme-programming
* http://www.extremeprogramming.org/rules/userstories.html
* http://scrummethodology.com/scrum-user-stories/
* http://wiki.servicenow.com/index.php?title=Well-Written\_Scrum\_Stories
* http://agilebench.com/blog/the-product-backlog-for-agile-teams
* http://improvement-services.nl/blog/?p=196
* http://www.extremeprogramming.org/rules/spike.html
* User Stories Applied: For Agile Software Development, by Mike Cohn, Published Mar 2004 On Empirical Research into Scrum, by Dr. Mark C. Paulk

**17. TEAM CHARTER**

|  |  |  |
| --- | --- | --- |
| **Course Title** | CPSC 544 Advanced Software Process | All team members participated in the creation of this charter and agree with its content. **Date** 08/30/2016 |
| **Instructor** | Dr. Chang-Hyun Jo |  |
| **Course Dates** | 08/22/2016 – 12/09/2016 |  |

**Team Members**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Address (city, state, country) | Phone | Cell | Email |
| Sree Harsha Ravilla | Fullerton, CA, U.S.A | 657-445-8111 | 657-445-8111 | sreeharsha@csu.fullerton.edu |
| Shivam Rana | Fullerton, CA, U.S.A | 714-725-2815 | 714-725-2815 | shivamrana@csu.fullerton.edu |
| Yamini gadidam | Fullerton, CA, U.S.A | 714-592-5098 | 714-592-5098 | yaminigds@csu.fullerton.edu |
| Ashish Paul Stephen | Fullerton,C.A, U.S.A | 818-468-7356 | 818-468-7356 | Ashish\_paul94@csu.fullerton.edu |

**Team Member Skill Inventory**

|  |  |
| --- | --- |
| Yamini | * MS Office * Desktop Publishing * Project Management * Android programming(Java) |
| Sree Harsha | * MS Office Suite (Word, Excel, PowerPoint, Project) * Front end web designing * Basic android application knowledge * Software engineering techniques |
| Ashish Paul | * MS Office Suite * Web Designing & Developing * Graphic and Visual Designing * Project Management |
| Shivam Rana | * MS Office * Software analysis and design * Software Lifecycles * Programming Languages (C#, Java) * Project management |

**Team Goals**

|  |
| --- |
| * Develop good relationship with teammates. * Put in the best efforts while adapting systematic and punctual approach. * Learn and teach from/to teammates. * Maintain dignity and integrity. * Develop skills to facilitate future career goals and industry. * Produce a good report and receive a good grade. |

**Team Roles**

|  |  |  |
| --- | --- | --- |
| Product owner | Sree Harsha | * Holds the vision for the product. * Create and prioritize the product backlog. * Choose the goals for each sprint. * Facilitate communication between the team and the stakeholders. * Ensure the team is building the right product. * Decides when to ship the product. * Planning the release. * Understand the value of the project. |
| Scrum master | Ashish Paul | * Track and ensure that the team follow the process * Help the team to move forward and be productive. * Coach the team. * Help everyone understand Scrum. * Manage the development process * Identify and eliminate obstacles that prevent the team from achieving their goals * Ensure a clear communication among everyone involved in the project. |
| Development team | Yamini | * Builds the product. * Deliver each increment of the product. * Development to achieve sprint goals. * Implementing test cases. * Unit and initial Acceptance testing. * Identify obstacles and informing the Scrum Master. * Self-organizing. * Daily Scrum meetings. |
| Shivam Rana |

**Ground Rules**

|  |
| --- |
| * All team members must be punctual and prepared for each team meeting. * The team will meet according to the necessity of the hour either via chat or conference call to discuss current and upcoming projects or assignments. * Team members will notify the lead in advance if they are not going to be able to attend a scheduled meeting. * Notify team of emergencies that may result in not being able to meet deadlines or meetings. The rest of the team will do their best to pitch in on the team assignment. * Participation and input is expected from all team members.  All opinions will be considered and equally valued. * The team must maintain open, clear, and effective communication at all times. * Maintain a positive, honest, and open atmosphere by respecting other members’ suggestions, using constructive criticism, and encouragement. * All team members will be held accountable for their portions of the projects and are expected to complete them in a timely manner and doing the best job they can. * Assist fellow team members when they are in need. * No plagiarism. Every member must be responsible to avoid/prevent plagiarism. * Team members should reply to email within 24 hours. |

**Time Commitments/Availability** (Pacific Time)

|  |  |
| --- | --- |
| Sree harsha | * M&W - 10am-3pm * T&Th - 3pm-8pm * F - All day * S&Sun - Can make arrangements if necessary |
| Ashish paul | * M&W - 10am-3pm * T&Th - 2pm-7pm * F – Full day * S&Sun - Can make arrangements if necessary |
| Shivam rana | * M&W - 10am-3pm * T&Th – 2pm-7pm * F – All day * S&Sun - Can make arrangements if necessary |
| Yamini | * M&W - 10am-4pm * T&Th - 2pm-8pm * F – All day * S&Sun - Can make arrangements if necessary |

**Conflict Management**

|  |
| --- |
| * In order to avoid conflict clear roles and responsibilities must be assigned, so that there is no confusion. * If conflicts arise, please bring them up to the whole team so that everybody can help to resolve the issue in a peaceful and harmonious manner. * If a team member is not performing, the team lead will speak to the member and try and resolve the issue. * The decisions over the conflict will be based on the majority votes of the members. * All team members must settle conflicts within the group as quickly as possible. |

**Risk Management**

|  |
| --- |
| * Emergency absence by the team members. It may be covered by team member with the least work load at that point of time and may get help from other members. * Team members’ non-availability in the meeting sessions should be by a makeup session from the recorder to know the facts. * Losing the data, assignments or important details would be avoided by maintaining backup. |

**Team Evaluation Criteria**

|  |
| --- |
| * Attendance for the meeting * Assignment completion * Punctuality in submissions and meeting time * Participation * Fulfilling duties of the role intended * Reliability and quality on the assigned work * Being in contact (text/email/call time of response record) * Behavior/attitude towards team members |

**18. TEAM EVALUATION**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Members** | Sree Harsha | Shivam Rana | Ashish Paul | Yamini | **Total** | **Comments on Your Evaluation on Team** |
| **Evaluators** |
| Sree harsha | 100 | 105 | 110 | 100 | **400** | I have evaluated the team on basis of work done but the individual and contributed towards the project |
| Shivam Rana | 100 | 100 | 90 | 100 | **400** | Appreciate the team work and team mates. |
| Ashish Paul | 100 | 95 | 100 | 90 | **400** | My evaluation was based on the amount of work each person did and how well did that person follow the time schedule. |
| Yamini | 100 | 100 | 100 | 110 | **400** |  |
| **Total** | **400** | **400** | **400** | **400** | 1600 |  |
| **Max Obtained** | 100 | 105 | 110 | 110 |  |  |
| **Average** | 100 | 100 | 100 | 100 |  |  |
| **Percent** | 100 | 100 | 100 | 100 |  |  |
| **Signature** | SH | SR | AP | YA |  |  |
| **Comments on Your Score Earned from Team** | It is fair grade | I have done the work assigned to me |  | Except for one evaluation, I think I have received a fair score |  |  |