HOSPITAL EMERGENCY MANAGEMENT SYSTEM

Hospital Emergency Management System

Project Report

Group Member's
Name

Contents

1. PROJECT CHARTER

Hospital emergency service provides treatment to those in need of urgent medical care. A hospital emergency room uses the strategy:

- Opting to help someone with a potentially fatal wound before someone with a less threatening wound
- Regardless of who arrived first
- Priority based service
  - Priority is given to those patients who need immediate medical attention.

1.1 PROBLEM STATEMENT

Develop an automated system to manage emergencies based on to the severity of the patient's medical condition. This problem will be solved using priority circular queues. Priority circular queue will priorities the patients according to their condition regardless of their arrival in the emergency.

1.2 CUSTOMER REQUIREMENTS

The system should fulfill the following requirements:

- Hospital Emergency Management System (HEMS) takes input from the patient based on the level of pain or severity of medical condition.
- This input is taken on the magnitude of 1 to 10, where 10 is the lowest level and 1 is the extreme level of pain.
- Priority of the treatment should be assigned based on the magnitude of pain defined by the patient.
- The order in which patients are treated comes from the following rules:
  - A patient of higher priority is treated before any patient of lower priority.
  - Two patients with the same priority are treated according to the order in which they were added to the queue.
- The system should be able to write the patient's record in file and save it.
- The system should be able to read patient's record from the file.

1.2.1 Functional Requirements

- System should be able to take input from the patient in the form of magnitude of the pain.
- Assign a priority to the patient according to the level of pain.

- The patient with the highest priority should be called first, regardless of who arrived first.
- Reading and writing patient's records from the file.

1.2.2 Non-Functional Requirements

- System should be efficient and fast.
- System should not consume more than 50MB of disk space.
- System should be able to run on minimum memory requirements.
- System should be able to run on both Windows and LINUX.
- User Interface of the system must be on command line.

To fulfill non-functional and functional requirements, this priority based system will be implemented using priority circular queues as:

- Circular queue are more flexible than linear queue
- Low computational time
- Memory efficient
- Priority assignment to each patient
- High priority patients treated first

2. PROJECT PLAN

2.1 DEVELOPMENT METHODOLOGY

Agile development methodology considers software as the most important entity and accepts user requirement changes. Agile advocates that we should accept changes and deliver the same in small releases. Agile accepts change as a norm and encourages constant feedback from the end user. Below figure shows how Agile differs in principles from traditional methodologies.

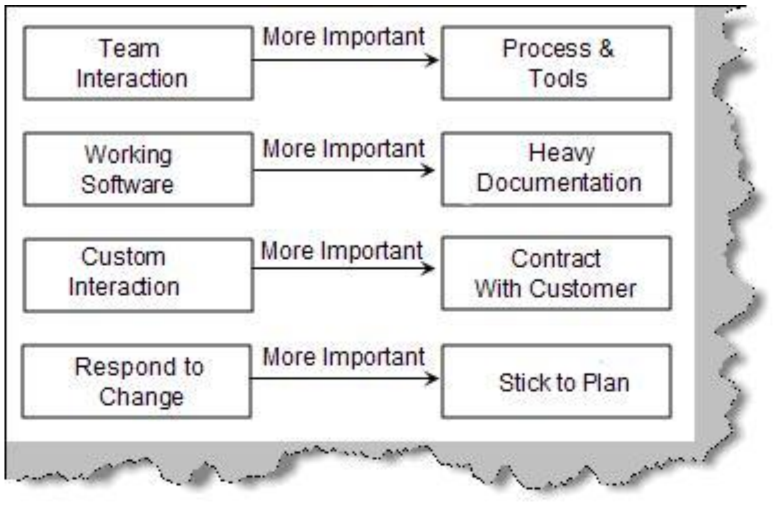


Figure 1: Change of Agile thinking

Principles of Agile Methodology

- Welcome change and adapt to changing requirements.
- Working software is the main measure of progress.
- Customer satisfaction is the most important thing and that can be attained by rapid, continuous delivery of useful software
- Day to day meetings between business people and development team is a must.
- Business and developers must work together. Face to face to communication is the most important thing.
- Deliver and update software regularly. In Agile we do not deliver software in one go, but rather we deliver frequently and deliver the important features first.
- Build projects around teams of motivated and trustful people.
- Design and execution should be kept simple.
- Strive for technical excellence in design and execution.
- Allow team to organize themselves.

Different Agile Methodologies:

Agile is a thinking approach to software development which promises to remove the issues we had with traditional waterfall methodology. In order to implement Agile practically in projects we have various methodologies. Below figure 'Agile Methodologies' shows the same in more detailed manner.

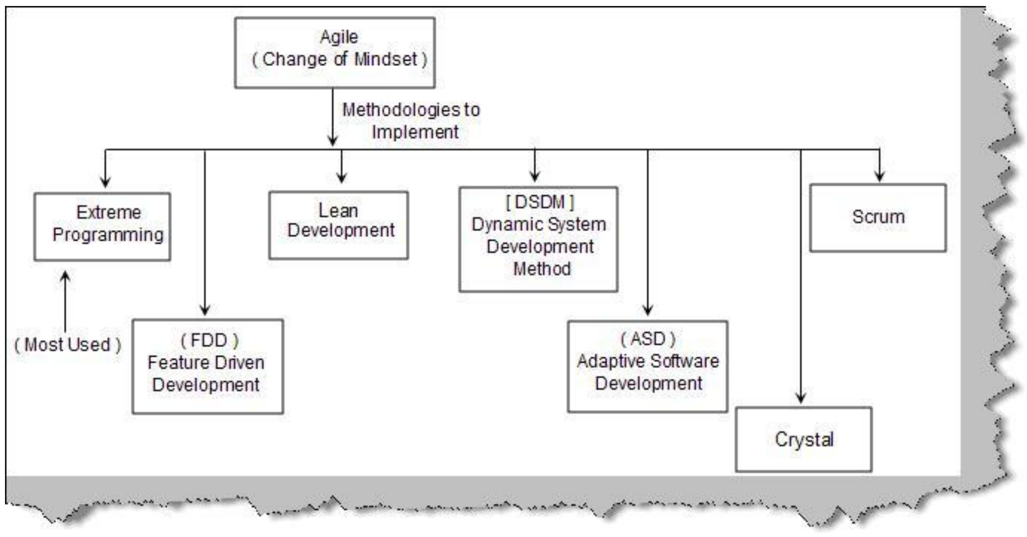

Agile
(Change of Mindset)
Methodologies to Implement
Extreme Programming
(Most Used)
(FDD) Feature Driven Development
Lean Development
[DSDM] Dynamic System Development Method
(ASD) Adaptive Software Development
Scrum
Crystal

Figure 2: Agile Methodologies

Among the various methodologies shown in Figure 2 we have chosen Extreme Programming because of the following reasons:

Extreme Programming (also termed as XP) is an agile software development methodology. XP focuses on coding of the software. XP has four core values and fourteen principles.

XP has four core values:-

• **Communication: -** Team should communicate on a regular basis, share information, discuss solutions and so on. Teams who communicate very often are able to solve problems more efficiently. For instance any kind of issues which are resolved in a cryptic fashion send an email to the whole team. This ensures that knowledge is shared with everyone and in your absence some other developer can solve the problem.

• **Simplicity: -** Keep things simple. Either it's from a process angle, technical angle or from a documentation point of view. An over complicated process or a technical architecture is only calling for problems.

• **Feedback:** - Regular feedback from end user helps us to keep the project on track. So egular feedbacks should be enabled from end user and testing team.

• **Courage: -** To bring change or to try something new, needs courage. When you try to bring change in an organization you are faced with huge resistance. Especially when your company is following traditional methodologies applying XP will always be resisted.

XP 14 Principles:-
From the above four core values 14 principles are derived which are as follows:

• **Rapid feedbacks:** - Developers should receive rapid feedbacks from the end user. This avoids confusion in the last minute of delivery. In water fall model feedbacks are received in late intervals. This is minimized in XP.

• **Keep it Simple: -** Encourage simplicity in project design and process. For instance rather than using complex tools probably simple handwritten flowcharts on board can solve the problem.

• **Give incremental changes: -** Whenever you update patches and updates, release it in small pieces

• **Embrace Change: -** Do not be rigid with the customer saying that we have already signed the requirement so we cannot change the architecture. End customer can change as the project moves ahead.

• **Light Weight: -** Keep documentation and process as simple as possible. Do not overdose the developer with unnecessary documentation.

• **Deliver Quality: -** Any code you deliver should be defect free. Be committed to your work and deliver defect free code.

• **Start small and grow big: -** Many times the end customer wants to start with a big bang theory. He can start with a big team, wants all the functionalities at the first roll out and so on. Start with small team and the "must have" features to be delivered. As we add features and the work load increases gradually increase your team strength.

• **Play to win: -** Take all steps which are needed to make a project success. Any type of deadline and commitment try to meet the same with true spirit.

• **Encourage honest communication: -** Promote honest communication. If communication happens face to face then there is less leakage of requirement. Encourage end user to sit with developers and give feedbacks; this makes your project stronger.

• **Conduct testing honestly: -** Test plans should not be created for the sake of creation. Test plan should prove actually that you are on track record.

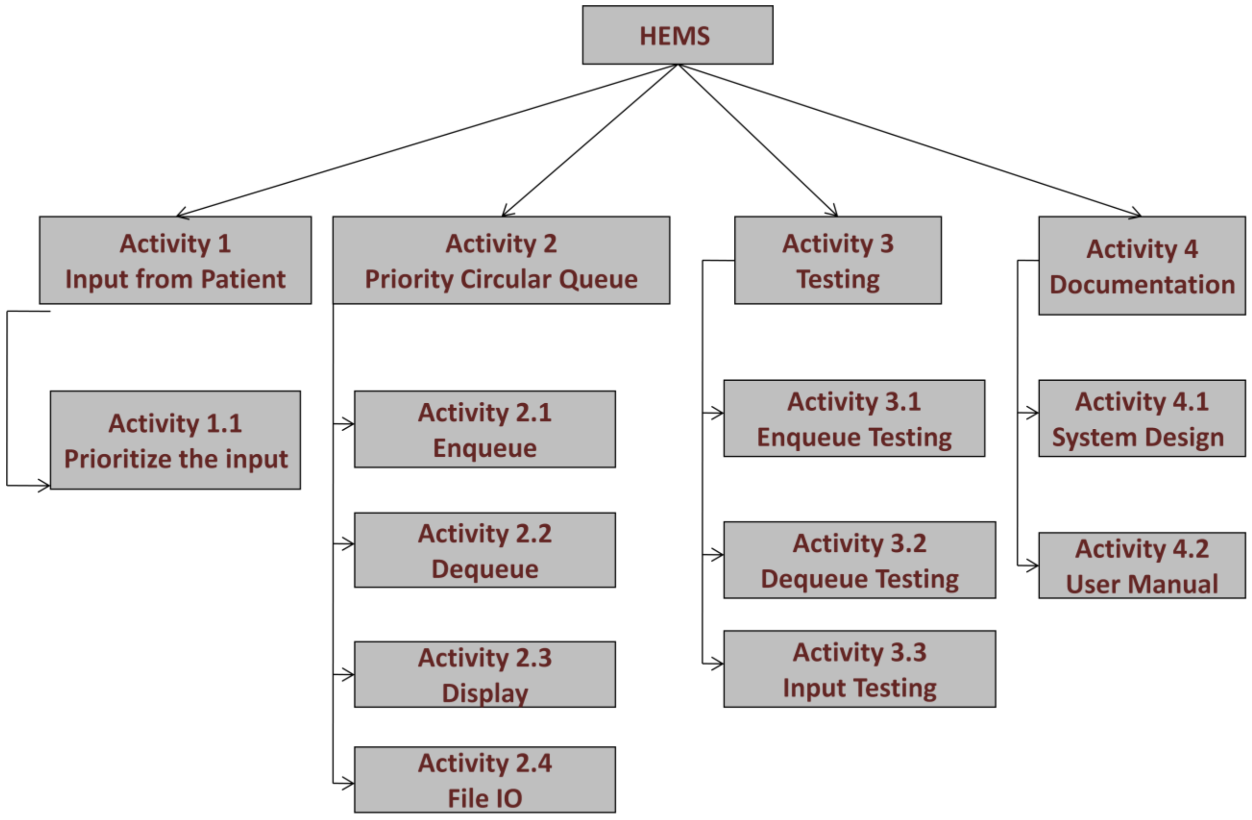
• **Adapt according to situation: -** No two projects are same, no two organization are same and behavior of people from person to person. So it's very essential that our approach also adapts according to situations.

• **Metric honesty: -** Pick metrics which makes sense to your project and helps you measure your project health.

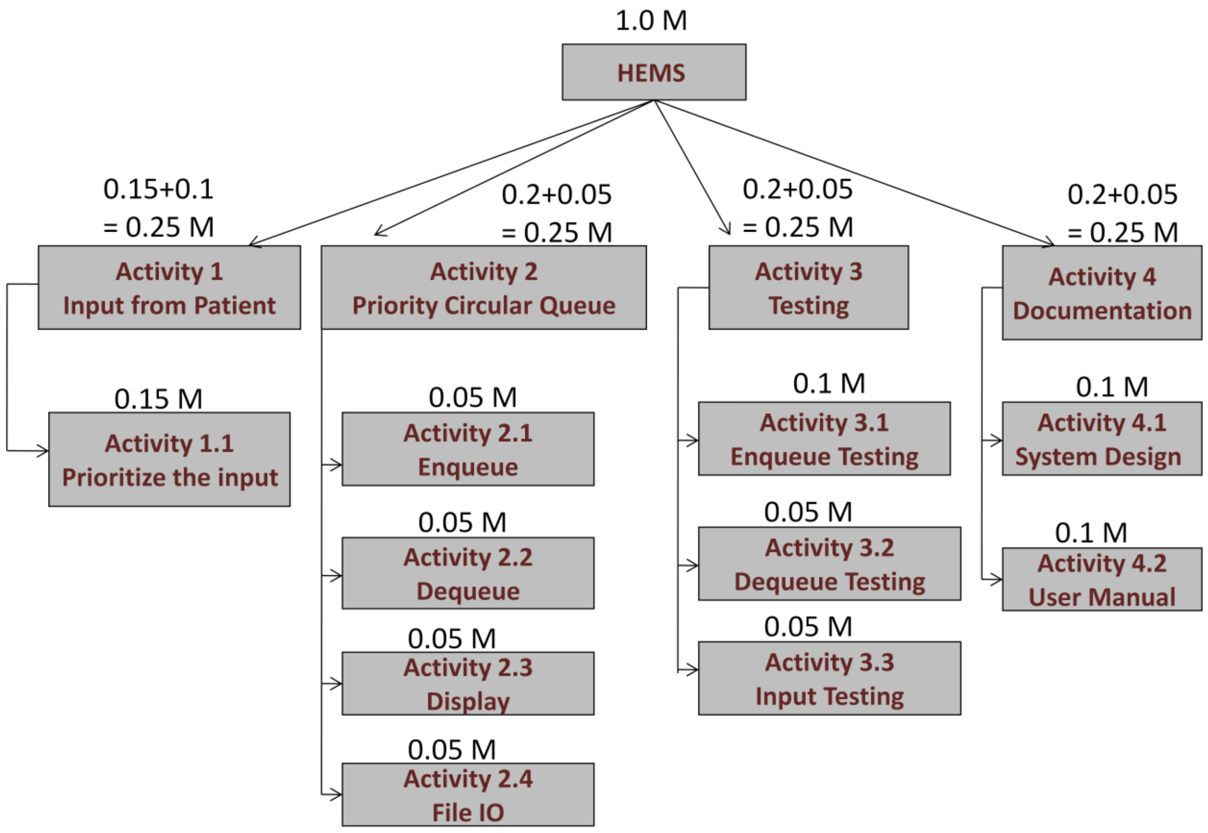
• **Accept responsibility: -** Do not impose or assign people on task which they do not like. Rather question the resource once which tasks he likes and assign accordingly. This will increase productivity to a huge level and maintains your project enthusiasm high.

• **Work with people's instincts: -** Normally in a project team there are highly motivated people, moderately motivated and people with less motivation. So give power to your motivated team members and encourage them.

2.2 WORK BREAKDOWN STRUCTURE


HEMS
Activity 1 Input from Patient
Activity 1.1 Prioritize the input
Activity 2 Priority Circular Queue
Activity 2.1 Enqueue
Activity 2.2 Dequeue
Activity 2.3 Display
Activity 2.4 File IO
Activity 3 Testing
Activity 3.1 Enqueue Testing
Activity 3.2 Dequeue Testing
Activity 3.3 Input Testing
Activity 4 Documentation
Activity 4.1 System Design
Activity 4.2 User Manual

2.3 COST BREAKDOWN STRUCTURE



2.4 PROJECT SCHEDULE

Work packages, Milestones and Timelines for four weeks						
	Work Packages	Tasks	Week 1	Week 2	Week 3	Week 4
1	Literature Review	• Review of Emergency System at Hospitals • Review of Priority Circular Queues • Documentation	▬			
2	Requirements gathering and Designing	• Gathering requirements from the customers • Selecting software engineering model to be adopted during the development • Documenting architectural and design aspects of the system • Status review meetings • Documentation	▬	▬◆		
3	Implementation	• Pre-processing the input to the system • Implementing Enqueue function for priority circular queue • Implementing Dequeue function for priority circular queue • File IO operations • Documentation		▬	▬◆	
4	Testing	• Writing test plan • Unit testing and integration testing • System testing • Acceptance testing • Documentation	▬	▬	▬◆	
5	Documentation	• Integration of documents • User Manuals				▬◆
6	Project Closing	• Handling over the project to the customer				▬◆

Milestone	◆
Progress	▬

2.5 RESPONSIBILITY ASSIGNMENT MATRIX

Responsibility Assignment Matrix

	Name	Responsibility
1	Member 1	• Describing Architectural View of the System • Pre-processing the input of the System • File IO implementation and its unit testing • Writing Test Plan • Compiling project Report
2	Member 2	• Describing the Design View of the System • File IO implementation and its unit testing • Integrating the System • System Testing • Compiling Project Report
3	Member 3	• Requirements Gathering • Implementing Enqueue • Unit Testing • Integration Testing • Writing User Manuals
4	Member 4	• Requirements Gathering • Implementing Dequeue • Unit Testing • Writing User Manuals
5	Member 5	• Writing Project Charter and Functional Specification • Implementing UI • Unit Testing • Review Status Reports • Project Closing Documentation

3. PROJECT EXECUTION

Two major issues were face while executing the program. These are given below:

Sr. No	Issue	Description	Cause	Solution	Status
1.	Priority Dequeue	Abnormal behavior while dequeuing patient from the queue	Invalid pointer assignment	Debugged the program and found the invalid pointer assignment.	**Issue Resolved**
2.	File Read Operation	System was able to read only one record from the file	New line character was not identified	Added a new line character at the end of each record	**Issue Resolved**

4. PROJECT MONITORING & CONTROL

Project monitoring is performed by reviewing the progress of the project. Project review has been done by conducting weekly meetings and expected and actual completion period of individual objectives were observed and documented.

Project objectives	Expected Completion Period	Actual Completion Period
1 Literature Review		
1.1 Review of Emergency System at Hospitals	1 day	1 day
1.2 Review of Priority Circular Queues	1 day	1 day
1.3 Documentation	1 day	2 days
2 Requirements gathering and Designing		
2.1 Gathering requirements from the customers	1 day	2 day
2.2 Selecting software engineering model to be adopted during the development	3 days	4 days
2.3 Documenting architectural and design aspects of the system	4 days	5 days
2.4 Status review meetings	1 days	1 days
2.5 Documentation	3 days	3 days
3 Implementation		
3.1 Pre-processing the input to the system	2 days	2 days
3.2 Implementing Enqueue function for priority circular queue	2 days	3 days
3.3 Implementing Dequeue function for priority circular queue	4 days	4 days
3.4 File IO operations	1 days	2 days
3.5 Documentation	3 days	3 days
4 Testing		
4.1 Writing test plan	4 days	5 days
4.2 Unit testing and integration testing	5 days	5 days
4.3 System testing	4 days	5 days
4.4 Acceptance testing	2 days	4 days
4.5 Documentation	5 days	5 days
5 Documentation		
5.1 Integration of documents	3 days	4 days
5.2 User Manuals	4 days	4 days
6 Project Closing	2 days	1 day

5. PROJECT TESTING

Project testing has been divided into unit testing and system integration testing.

5.1 UNIT TESTING

The system has following Unit Test Cases:

1. REQUIREMENT 1:

1.1	Testing Input from the Patient
Purpose:	To test the input from the patient. Patient must enter his/her pain level from 1-10, where 10 is the lowest and 1 is the extreme level.
Pre-Requisite:	System is running and input option is selected.
Test Data:	Pain level = {<1, 1, >1, <10, 10, >10, any character value, empty}
Steps:	• Run the system. • Press E, to start entering the patients by entering their pain level. • On successful insertion, patient is added in the emergency list. • If some invalid input is entered, the system will give error message .
Status:	

1.2	Testing Conversion of User Input to Priority
Purpose:	To test the conversion of User input (pain level) to Priority.
Pre-Requisite:	System is running, input option is selected and the user has entered a valid pain level.
Test Data:	Pain level = {1, 10}
Steps:	• Convert pain level from 1-10 in to priority from 1-10 • Priority 1 means critical patient, must be served first • Priority 10 means less critical patient, can wait for the turn
Status:	Success

2. REQUIREMENT 2:

2.1	Testing Priority Call-out of the Patients
Purpose:	To test the order in which patients are treated.
Pre-Requisite:	System is running, each patient has been assigned a priority and next patient to be served is called out using option'd' on the menu.
Test Data:	Priority Queue = {A patient with high priority is entered before patient with low priority, A patient with low priority is entered before patient with high priority, two or more patients have same priority }
Steps:	• Add patient to the queue with highest priority • Call out the patient • Highest priority patient called, regardless of its position in the queue • Two patients with the same priority are treated according to the order in which they were added to the queue.
Status:	Success

3. REQUIREMENT 3:

3.1	Testing File Write Operations
Purpose:	To test the file’s write operations. The patient’s records in the queue must be saved in the file when the system is quitted.
Pre-Requisite:	System is running and there is at least one patient in the queue.
Test Data:	File Write = {Queue is empty, Queue has only one patient record, queue has n-number of patients’ records}
Steps:	• Read one patient data from the queue. • Write/append the patient data in the file.
Status:	Success

3.2	Testing File Read Operations
Purpose:	To test the file’s read operations. The patient’s records in the stored file must be read and entered into the queue when the system starts running.
Pre-Requisite:	System is running and the stored file exists on the desired locations.
Test Data:	File Read = {File is empty, File has only one patient record, File has n-number of patients’ records}
Steps:	• Read one patient data from the File. • Add the patient data in the Queue.
Status:	Success

3.3	Testing File Location
Purpose:	To test the file’s Location on the disk. The file containing patients’ data is by default located in the working folder of the system.
Pre-Requisite:	System is running and file IO operation is requested.
Test Data:	File Location = {File does not exist, File is present on the default location, file is corrupted}
Steps:	• File IO operation requested. • File is not present on its default location, give error. • File is present on its desired locations, perform IO operation. • File is corrupted, give error.
Status:	Success

4. REQUIREMENT 4:

4.1	Testing Patient's Data
Purpose:	To test the patient's data. Patient's Name, NHS number and pain level is recorded in the queue while entering a patient.
Pre-Requisite:	System is running and enter patient option is selected.
Test Data:	Patient Data = { (Patient's Name, NHS number and pain level), any other combination of the three data values other than one specified before}
Steps:	• Take patient's name from the user. • Take NHS number; verify it (Test Case No. 4.2). • Take pain level, verify it (Test Case No. 1.1) and convert it into priority (Test Case No. 1.2).
Status:	Success

4.2	Testing Patient's NHS Number
Purpose:	To test the patient's NHS number. NHS number should not be more than 4 digits and no two patients have same NHS number.
Pre-Requisite:	System is running and enter patient option is selected.
Test Data:	Patient NHS Number = { NHS number >4, NHS number = 4, NHS number < 4, two patients having same NHS number}
Steps:	• NHS number less than 4 digits, give error message • NHS number greater than 4 digits, give error message • NHS number has 4 digits, ask for next input. • Duplicate NHS numbers, give error.
Status:	Success

5. REQUIREMENT 5:

5.1	Testing User Interface of the System
Purpose:	To test the UI of the system. Each option in the system is represented by a character.
Pre-Requisite:	System is running.
Test Data:	Option = { e, d, f, s, q, any other character than the ones mentioned earlier}
Steps:	• Option E. • Option D • Option F • Option S • Option Q • Any other character, give error.
Status:	Success

5.2 INTEGRATION TESTING

5.2.1 Entry Criteria

All functions have been unit tested.

5.2.2 Elements to be integrated

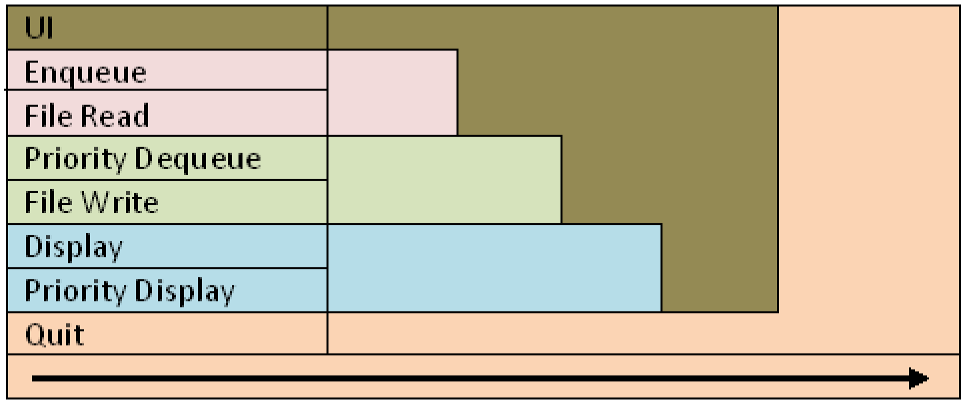
UI, Enqueue function, File Read, Priority Dequeue function, File write, Display, Priority Display, Quit.

5.2.3 Integration Strategy

Bottom-up strategy is used to integrate the modules. The reasons for selecting this strategy is that interface errors are detected earlier, confidence in system, errors in critical modules are detected and major functions and processing are tested earlier.

5.2.4 Software Integration Sequence

Given below is the integration sequence plan in which different colors show the sequence in which the functions are integrated.



6. PROJECT CLOSING

The project is closed when the customer executes the acceptance test plan and the product meets the customer's requirements. At project closure, the customer has to sign the customer acceptance form so that the project can be formally closed.

Customer Acceptance Form

Hospital Emergency Management System

Version 1.0 Issued 29 November 2010

Project: Hospital Emergency Management System	
This document has been issued by: Project Team	Date Issued: 31/08/10

The Project Outcome has been measured against its acceptance criteria and has been formally accepted on behalf of the customer. The project may now be closed.

Additional Comments about the Customers Acceptance:
Recorded Shortfalls of the Final Project Outcome (if any):

Executive / Sponsor:	Signature:____________________
	Name:___________________
	Date: ___________________
Project Manager:	Signature:____________________
	Name: ___________________
	Date: ___________________