|  |  |  |
| --- | --- | --- |
|  | Document Number |  |
| Based on Template |  |
| Created By |  |

Jampp- Clientify your Life

Software Design Specification

This document describes the design and implementation of Jampp- Clientify your Life.

Reviewers

|  |  |
| --- | --- |
| Department | Name/Title |
| Development Engineering |  |
|  |  |
|  |  |
|  |  |

Table of Contents

# Purpose

The purpose of this document to provide detailed design of the project based on client server distributed computing- **Jampp- Clientify your Life.**

# Design Block

<>.

USER

SERVER

CLIENT

CLIENT

CLIENT

CLIENT

CLIENT

# Tasks:-

|  |  |
| --- | --- |
| **TASK** | **STATUS** |
| 1Define groups (ports) |  |
| 2 Create FSM for each group (2 states) |  |
| 3 Listener code (6 ports, bidirectinal, queue) |  |
| 4 Message and struct definition |  |
| 5 Message handlers(5) |  |
| 6 Client code (fsm,1 port, bidirectional,alive) |  |
| 7 CLI |  |
| 8 group/client table, client alive table, Data table(5 tables) |  |
| 9 Debuggability |  |
| 10 handling of keep alive message and killing of client with timeout |  |

**DIFFERENT FILES TO HAVE IN THE CODEBASE:-**

Files and tasks to be done are given below:

/server

1 server\_init.c

2server\_cli.c

3msg\_def.h

4msg\_def.c

5server\_main.c

6server\_fsm.h

/client:

1 client.c

2 msg\_def.h

3 msg\_def.c

4 client\_fsm.h

**TASK 4 AND 5:-**

Details of structs to be used:

enum:

INITIAL\_JOIN\_REQ

INITIAL\_JOIN\_REPLY

GROUP\_JOIN\_REQ

GROUP\_JOIN\_REPLY

KEEP\_ALIVE

DATA

struct initalJoinReq {

int msg\_type;

}

struct initialJoinreply{

int msg\_type

int clientId

stuct groupList \*GL;

}

struct groupList {

int groupId;

int port

}

struct groupJoinReq {

int msg\_type;

int clienId;

bool status;

}

struct keepAlive {

int msg\_type;

int clientId;

}

struct data {

int msgType

int clientId

int groupId

char \*data;

}

# Functional Description

This is for the phase 1 implementation. This handles the client server code including the FSM.

Client is responsible to send a request to the server and join the group.

Server is responsible to

* listen to the request,
* send the available groups to join to the client.
* Maintain the mcast group table.

## Server- Client code

<>.

## Server side FSM

<>.

## Client Side FSM

<>.

## CLI Support

## Debuggability

# <>

.

# Branching Strategy

Git repository. We have a master branch and a main child branch “staging”.

Dev branch is pulled for all development activities. Child of dev branches to be pulled by each developer to commit the code.

Every week stable code to be moved to the dev branch.

Every month the code is moved to the staging branch.

Final commit/Demo/Release to happen from the master branch.

Master

Staging

Devloper code

Devloper code

Devloper code

Devloper code

Devloper code

Dev

Demo

Demo

Demo

# Reference Document