System Class Requirements for Project 1

Project Description:
Project is the implementation of an interpreter that will process Dalvik bytecode. The interpreter is to be written in Javascript and supported by a common browser. The output of the interpreter should execute Javascript code.

A number of Dalvik Op Codes are defined and listed here:
http://www.netmite.com/android/mydroid/dalvik/docs/dalvik-bytecode.html

System Requirements:

1) The system shall provide the capability to read in a file that contains bytecode data
2) The system shall parse the bytecode data
3) The system shall map the dalvik syntax to javscript syntax
4) The system shall produce an output of results
5) A project report must be created that is no more than 5 pages
6) A project package must be created that includes instructions for (Setup, Build, Run and Test), Source and Binary code.
7) The customer will provide details of acceptance test before final due date
8) Project completion date is approximately four weeks from start Sept 23, 2012

Software Requirements:
1) FireFox 15.0.X or higher will be the common browser that will run the interpreter code
2) A Linux environment will be used to develop, test and perform the user acceptance test
3) A web server shall be supported to host the HTML and Interpreter code
4) The Web page shall support a file upload mechanism
5) A list of Dalvik Op Codes that will be supported is shown below:

Instruction	Description	OpCode	Supported
move	Register to register move	01 -0d	
move-result	Move a method result to a given register		Limited No truncating bits
Move-exception	Move exception to a register		No
return	if no value, void	0e – 11	
const	Move a constant into register dest	12 – 1c	
Monitor-enter		1d -1e	No
Monitor-exit			
Check-cast	throw ClassCastException if src cannot be type	1f	No
Instance-of	Set dest to 1 or 0 according to whether src is type	20	
Array-length		21	
New-instance		22	No
New-array	see filled-new-array for argument registers?	23-26	No
Fill-array	put items from data into array at dest		
throw	throw an exception object	27	
goto	jump to a different place within the method	28 -2a	
switch	execute a switch statement		
Packet switch		2b	No
Sparse switch		2c	No
cmp	primitive_type may be float, double, or long	2d	
If		32-38	
If-eq	If (a == b) goto address		
If-ne			
If-lt			
If-ge			
If-gt			
If-lt			
If-eqz	if(src == 0) goto address		
If-nez			
If-ltz			
If-gez			
If-gtz			
If-ltz			
Array		44-51	
Array-get	Primtype = { int, wide, object, boolean, byte, char, short }		
Array-put			
instance		52-5f	
Instance-get			
Instance-put			
Static		60 -72	
Static-get			
Static-put			
invoke	Call a method of kind = {virtual, super, direct, static, interface}	74 – 78	
negative	primtype = { int, long, byte, float double }		
Primitive-cast	src/destType = {int, long, float, double}	7b – 8f	No
Int-cast	destType={ byte, char, short }		
Math functions		90 -af	
add	type={int, long, float, double}		
sub	type={int, long, float, double}		
mul	type={int, long, float, double}		
div	type={int, long, float, double}		
rem	type={int, long, float, double}		
and	type={int, long}		
or	type={int, long}		
xor	ints only		
shl	ints only		
shr	ints only		
ushr	ints only		
I,L,F,D/2 addr		b0 -cf	No
add	type={int, long, float, double}/2 addr		
sub	type={int, long, float, double}/2 addr		
mul	type={int, long, float, double}/2 addr		
div	type={int, long, float, double}/2 addr		
rem	type={int, long, float, double}/2 addr		
and	type={int, long, float, double}/2 addr		
or	type={int, long, float, double}/2 addr		
xor	type={int, long, float, double}/2 addr		
shl	type={int, long, float, double}/2 addr		
shr	type={int, long, float, double}/2 addr		
ushr	type={int, long, float, double}/2 addr		
Int/16 & int/8		d0 -e2	No

6) Software shall present results of interpreter processing

No functional requirements:

None Provided