Getting Your Feet Wet

Once upon a time, building your own software empire meant becoming another Microsoft or Bill Gates. And that took plenty of time, plenty of money, and plenty of resources.

Nowadays, it’s as simple as turning on your computer.

Well, maybe not THAT simple. But having a computer is really all it takes to own, operate, and develop a software business.

Just having access to the Internet…

- You can brainstorm and come up with ideas.
- You can find out what type of products do or don’t already exist.
- You can determine exactly what people want and need.
- You can locate and hire a programmer.
- You can create software graphics (or have them created).
- You can promote and sell your finished products.

And when you think about the fact that you can do all those things without leaving the comfort of your home, it’s pretty amazing. Plus, the cost of creating and developing software is no where near what it used to be.

Instead of having to travel long distances or settle for whatever programmer is located within a reasonable distance from you (and whatever price they happen to charge), you can easily choose from any number of qualified programmers throughout the world.

For each project you put out there, you’ll have numerous professionals vying for your business, each one trying to outbid the other. And in most instances, that means coming up with a bid that is lower than the next guy.

Not that you should pick the lowest bid. That should never be your main consideration when choosing a programmer. But the fact that programmers will be competing in that manner means you can ultimately get the best possible job done for the best possible price.

Another advantage is the income potential. Although there are several ways you can make money online, none of them comes close when you calculate the amount of money that can be generated through the sale of software products.

And the true benefit of dealing in software products is the fact that you don’t have to talk someone into buying it. The product either fulfills the needs or wants of a prospective buyer or it doesn’t.

With an ebook, for example, you generally have to come up with all sorts of ingenious sales copy just to convince people why they should buy it. With software, it’s as simple as listing all the features and benefits.

And, when it comes right down to it, people who purchase software products aren’t really interested in hearing some sales pitch. They simply want the facts and nothing but the facts…

- Does this product have all the features I want/need?
- Will this product help me do something faster/easier?
- Is this product easy to install?
- Is this product easy to use?
- Will this product work on my operating system?
- Does this product come with help documents?
- Is there any technical support?
- How much does this product cost?
- Will there be future upgrades and are they free?

Those are the type of questions that run through the mind of your typical software buyer. If you can deliver everything they're looking for at a fair price, you've made a sale. If not, they'll simply move on to one of your competitors.

In order to be successful in the software industry, you need to be constantly aware of what people want and need. But being aware is only the beginning. It's also necessary to create quality products that will satisfy any and all of the buying public's requirements. And in many instances, greatly exceed their expectations.

Notice, however, that I used the word "quality" with regard to creating products.

That's the one thing that's absolutely imperative in this business. If you merely run around producing second-rate clones of existing products, or you develop products that don't work properly, you might as well pack it in.

There's no way you'll make it in this business doing things that way. There are tens of thousands of people creating and developing new software products. And there are millions of products being sold. Some good, some not.

In order to compete – in order to SUCCEED – you need to take this very seriously.

You need to come up with good solid quality products that will please the people who buy them.

If you can't do that – if your only objective is to get rich quick – you'll never survive.

And you'll certainly never make any decent money.

But let's not even go there.

Let's assume you're one of the "good guys", that you're hoping to develop your software business the right way. That you plan to deliver products that are innovative, unique, and possess the kind of quality John and Jane Q. Public are hoping for.

That being said…

From this point on, you’ll have benefit of all my knowledge and expertise with regard to making money with software products. Not based on conjecture or speculation, but through the course of actually creating and developing my own successful software business over the past several years.

It’s been fun, it’s been challenging, and of course, it’s been profitable. But most of all, I would have to say the entire journey has been thoroughly rewarding. Both personally and professionally.

Now it’s your turn…

Brainstorming

One of the most interesting aspects of creating and developing software is coming up with new and innovative ideas.

But that doesn't necessarily equate to creating an entirely different piece of software.

Oftentimes, it's simply a matter of taking something that already exists and making it better. Or expanding on a current program.

It's pretty much the same thing that inventors go through… They recognize that a product is the perfect candidate for improvement. Or, they envision an entirely new product, something that no one else has yet to develop.

That's how it is with software products. So you need to look around you… Take careful note of what's already out there and how it works.

Take note of what tasks you (or others) frequently perform.

Then ask yourself…

What software has room for improvement?

What software would make life (work) easier?

When it comes to "thinking up" software products, it's most often a case of simply being totally aware of what is taking place right in front of your eyes.

Is there a webmaster task that you currently perform manually? One that drains a considerable amount of valuable time? Talk to a programmer. They might very well be able to create a software program that can perform the task automatically.

Is there a void in a particular marketplace or genre that you feel should be filled? Talk to a programmer. They might very well be able to create that perfect financial or gaming software product.

For the most part, consumers fall into the following three categories…

- They have a particular problem that needs to be solved.
- They are looking for something that will make their life easier.
- They want to be entertained.

Overall, you should always be looking for some need or void and then create a software product that can satisfy it.

One fulfillment would be to create software that automates or simplifies a particular task involved in marketing. Take the concept of "tag and ping" for example… The moment it become widely known that social bookmarking sites and the various services they provided could be used as effective marketing techniques, the software wheels began to turn.

In no time at all, various products and programs surfaced…

- utilities that could automate the process of creating links that need to be associated with tags
- programs that automatically submit mass quantities of URL's to various bookmarking sites

- Plug-ins for existing blog software that can enhance or improve various aspects of posting, tagging, and pinging

Basically, every time a new marketing technique surfaces, there are software programs that can be created to either automate or enhance the technique.

Within my own products, RSS Equalizer (http://www.rssequalizer.com) is a good example of satisfying not one, but several marketing needs.

To begin with, webmasters needed fresh content on their websites in order to satisfy search engines. RSS feeds could provide that. The only problem was the fact that the content of most RSS feeds is somewhat restrictive.

If you have a sports site, for example, you can set up different feeds for different areas of interest… baseball, football, soccer, Nascar. But what if you have numerous pages in each of those categories. And each of those pages is optimized for a specific keyword. Then what?

Your only choice would be to either place the same RSS feeds on multiple pages or just restrict the feed to primary (or select) pages for each category. Unfortunately, the former would result in duplicate content and the latter would results in pages that had no fresh content.

That’s where my idea for RSS Equalizer came about…

Webmasters needed a way to place targeted RSS feeds on all their keyword optimized pages. In other words, the content of the RSS feeds needed to be based entirely on what keyword was being used.

RSS Equalizer does just that. It allows you to place RSS feeds on any web page and have the content of that feed specifically targeted toward the keyword for which the page is being optimized.

For the most part, that’s what the majority of these types of software products do.

They fill a void or need that happens to exists.

What you need to realize, however, is that the void or need in many instances might not be quickly and easily recognized. That’s why it’s important to always be alert, always be looking for that new angle or possibility.

And don’t assume that an idea might be too bizarre or unusual to qualify as a profitable software product. The only criteria that should ultimately matter is whether or not there’s a solid market potential.

Which brings us to another very important issue. You absolutely, positively MUST know your target audience.

If you’re going to create new gaming software, for example, you have to be knowledgeable about what types of programs (and characters) are most popular and what kind of interface and functions the users demand.

If you’re going to create a new ebook compiler, you need to know exactly what features the people who use that type of software would require. You also need to determine what features they themselves would add to that type of product if they were given the opportunity.

Without knowing what your target audience wants or requires, you’re just spitting into the wind, hoping you come up with something they want.

Don’t guess. Don’t assume. And don’t try to think for someone else. If you’re going to make money selling software, you have to be totally familiar with the people who will be purchasing your products. BEFORE you get in too deep.

Beyond that, the field of possibilities is wide open. And the number one rule is this… Don’t limit yourself in any way, shape, or form. If you can envision it, a programmer can most likely make it happen.

Researching The Market

No matter how good an idea or concept might be, it’s worth nothing if it doesn’t have the ability to generate income. In other words, if you can’t sell it, don’t build it.

But how do you determine just what types of software will sell and which ones won’t? And beyond that, how do you determine what products have the potential to earn you substantial amounts of money?

The truth is, it’s not always easy. And sometimes – even after you’ve conducted all sorts of research and taken numerous surveys – you still don’t have a definitive answer.

All you can do in the end is go with your gut instinct.

Okay. So maybe you don’t want to risk your entire life savings on gut instinct. Fair enough…

Although there will always be some risk involved (you can never be one hundred percent certain of any software product), you can certainly narrow things down to where the odds of financial success are in your favor.

The first thing you need to do is find out whether or not there are any existing products that are either the same or similar to what you have in mind.

The easiest way to accomplish that is to simply conduct searches using terms that would accurately describe the type of software or product you have in mind.

Although search engines would be a good place to start, you might get better and quicker results looking through the various download sites like CNET (http://download.com) and Tucows (http://www.tucows.com).

If you find any, you need to compare them to your own idea. In some instances, that could mean that you have to actually purchase the product so you can personally use and evaluate it.

Of course, if a demo or free trial is available, you could simply download and review that version. Some of the features might be turned off, but you’ll still be able to see which ones are included in the software.

Then ask yourself…

Will your own ideas improve on what already exists? Will they add overall value or contribute something new or original to the use or functionality of the software? In general, will your ideas make the product better?

If the answer is yes, go ahead and explore any and all improvement and upgrade possibilities. If the answer is no, then you need to come up with some other product.

But even if you initially find that no other software products like yours exist, don’t be too quick to jump and shout. There might be a very good reason no one else has created it.

And the most blaring reason might be the fact that it’s a poor product choice, that there’s no probable market or profit potential.

Or, it could simply mean that no one else has yet thought of that particular software product. If that’s truly the case, then the marketing and profit potential is wide open.

That’s not to say it’s a slam dunk. It’s not. It just means that you’ve got a good candidate for a software product. You still need to establish whether or not it will actually sell.

Which brings us to your typical “catch 22”. You don’t want to create software until you’re confident it will sell. But there’s no way of positively knowing what purchase power a product has until you actually put it out there.

That’s where gut instinct comes into play…

If you’ve thoroughly researched your market, and you have a pretty good handle on what the people in your target audience want, you should already be confident about your prospective software product.

You have to understand that it’s not about creating and selling software products. It’s about knowing what software products to create. And there’s only one way to come up with that kind of knowledge.

You need to have your finger on the pulse of your target audience. You have to climb into their skin and feel what it’s like to be them. And once you’ve done that, you should be able to answer the questions that will determine what software products should be created.

So it all comes down to one important factor…

You need to be able to choose software products that are most likely to satisfy the needs and wants of a particular group of individuals. And by the time you’re finished researching and studying any given group, you should certainly have that ability.

And when your gut instinct is sending loud and clear messages to your brain… “this is it, this is it, this is it”… it’s definitely time to listen!

Establishing Features

Once your gut instinct kicks into gear – assuring you that your idea for a product has the right stuff to be a financial success – you can begin to establish specific details about the software itself.

Make a list of all the characteristics, features, and options you want your software product to have.

Include anything and everything you can think of. For example, if you were creating a software program for a personal membership site, part of your list might look something like this…

Unique username
Password encryption

Secure login
Lost password
Members can update their own profile
Option of hiding their email address from other members
Uploading member photos or videos
Public chat room
Free membership
Paid membership upgrade
Two levels of paid membership
Membership billing integrated with PayPal
Unsubscribe (delete membership) link within member profile
Affiliate program signup link within profile

What you're trying to do here is list all the things you want – or think you might want included in your software product. That way, the programmer (or prospective programmer) will have a solid concept of what you have in mind.

But don't just rely on your own thinking. In order to make certain you don't overlook something, you need to scrutinize other software products. They don't necessary have to be similar to yours (although it would help). You just need to get a clear indication of any and all possibilities.

For example, you might come across another product that gives their members the opportunity to contact one another within the membership area itself. That type of internal mailbox might not have been something you initially thought of, but it might very well be something you would like to offer your own members.

Or perhaps you encounter a totally different software product that you would like incorporated into yours. For example, a utility that generates customized pop-up windows. Rather than have your members access it independently, all they have to do is log in to their account and click on a link.

Or, you could create a toolbox area where your members could easily and quickly access a variety of utilities. And the advantage would be the fact that they are all built in to the primary membership program and not merely "attached" to it.

The whole idea of looking at other software products (aside from seeing what's currently out there), is to educate yourself with regard to what they offer. But the bonus is, looking at other products can frequently spark ideas that you might not have otherwise thought of.

A great place to access every conceivable type of software is Hot Scripts (http://www.hotscripts.com). Their database is quite extensive and includes programs and scripts in every popular programming language (something we'll discuss further on).

Once you go there, be prepared to spend a considerable amount of time. But whatever you do, don't dismiss this area of educating yourself. It's extremely important to know what's hot and what's not, both in terms of programming and user preference.

That's one of the other advantages of the HotScripts website. The users themselves have rated a good portion of the software products. That means you can see at a glance what level of quality each one carries.

Just browsing through the different categories and reading the descriptions, you can get a ton of ideas. Not only for enhancing or improving a software product you already have in mind, but for coming up with entirely different products.

Either way, the more browsing you do, the more features and details you'll be able to add to your list. And the more information you can provide a programmer, the more likely you are to get the exact product you're looking for.

Naturally, once you get farther along, you can discuss each and every aspect with a programmer. A good one will even make suggestions, helping you decide what features to include as well as what programming options exist with regard to individual features.

But browsing the web and places like HotScripts beforehand will give you a distinct edge. Not only in terms of how efficiently you can work with a programmer, but the end result as well.

Quite simply, the more knowledge and information you have about software products and their overall development, the greater the success you'll ultimately enjoy.

Loose Lips Sink Ships

Before we move on, I feel it's important to mention a pitfall that's relatively common among first-time software developers…

The one thing you should NEVER do is give away too much information prior to securing a programmer. Especially if you're hoping to develop some new and innovative software product.

Be aware that there is always someone paying close attention to what's happening in the world of software. If they catch wind of a good idea, they can just as easily create their own product based on the information they've gathered.

I understand that the desire to get someone else's approval or opinion is only natural. You have this idea but you're not a hundred percent certain about it's true potential. So you kind of throw it out there to see what sort of reaction you get.

In all honesty, this is the place where most people hoping to create software products oftentimes drop the ball. They feel the need to see what someone else thinks about their idea. But for someone wanting to create and sell software products, that can be deadly.

The ONE thing you need at this point is total self-confidence.

If you're constantly second-guessing yourself, if you're not sure about your ideas, if you have to ask someone else what they think before you can move ahead… you're going to have a very tough time getting any software product developed.

Before you open your mouth, ask yourself…

Just how well do you know the person you're talking to? Are you absolutely certain they won't "casually" mention it to someone else who might or might not be a person who can use the information to their benefit?

We're not talking about your immediate family here (unless, of course they happen to work for a company that deals in software development). What we're talking about is having loose lips, telling someone about your product idea without being absolutely certain that person won't ever mention it to anyone else.

You need to have total confidence in whatever ideas you come up with. And assuming you've conducted the proper research with regard to market, public interest, etc., that confidence should be rock solid.

Therefore…

If you have confidence in what you're doing, there's no reason to discuss it with anyone else. Or ask them what they think about it. Or prompt them to give you input with regard to whether or not it's even a good idea.

On the other hand…

If you're unsure of yourself, if you need someone else's input (or worse, approval) before you can move forward, you probably shouldn't be in the business of creating and developing software in the first place.

To be perfectly blunt, this ain't no place for wimps. You have to believe in yourself and have confidence in your ability to gain financial success.

Will you create the perfect product first time around? Maybe, maybe not.

Will you get rich off the first product you create? Probably not.

But if you're not willing to get out there and give it your best shot, then you're better off not getting involved in the first place.

There are no guarantees in software creation and development. The best you can do is conduct the proper amount of research to determine the probability of success and then make a qualified decision based on that information.

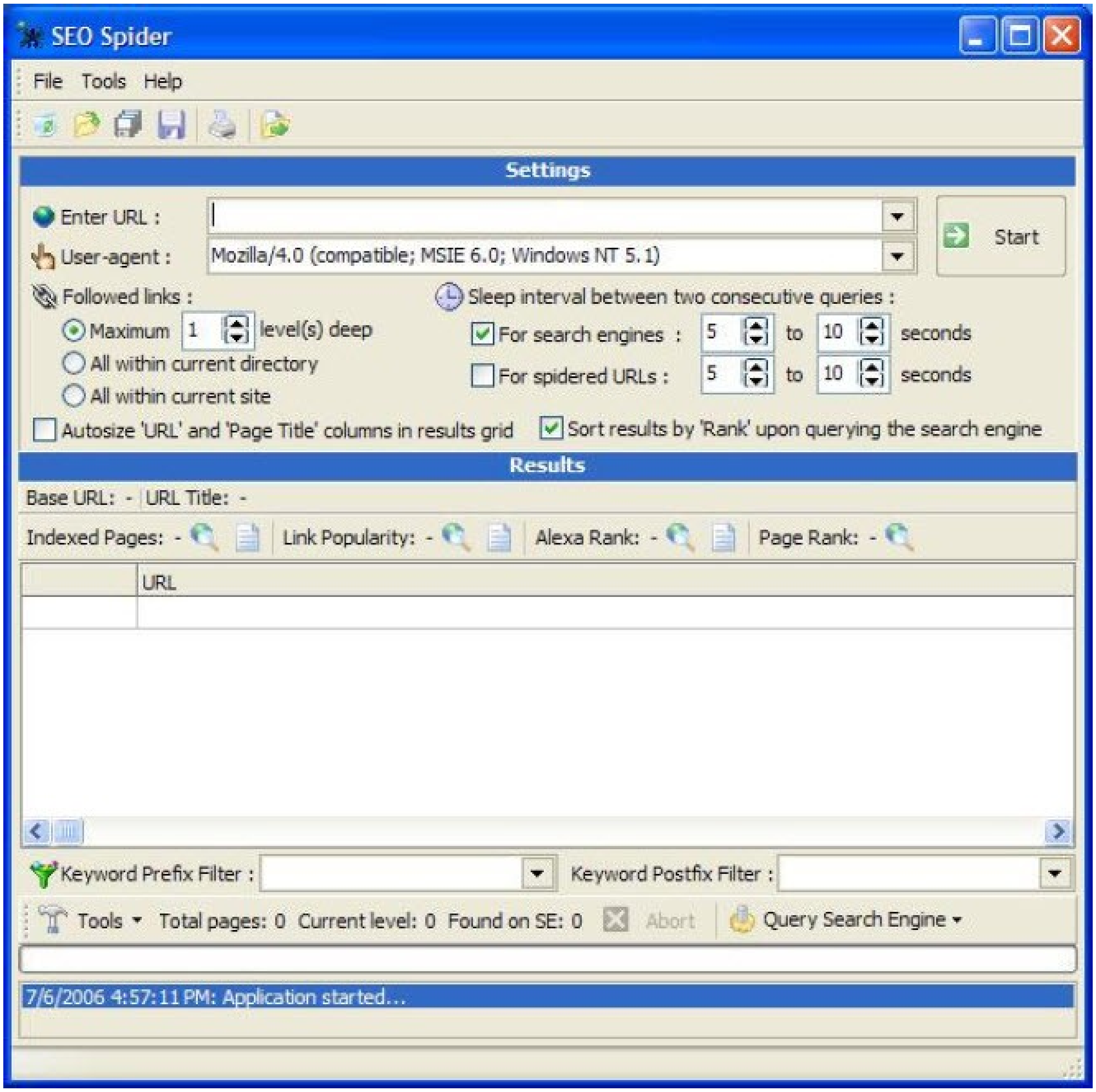
And whatever else you do, make absolutely certain you keep your "chatter" under control. NO loose lips!

Defining The Details

Whether it stands alone (like a computer game) or it runs in the background (like an anti-virus system), most software will display some sort of screen whenever the program is opened or used.

So that's one of the first details you need to decide upon. What type of screen (or user interface – UI) will the user first be confronted with?

Will it be the program itself…



Enter URL :
User-agent : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

Followed links :
[✓ Maximum] 1 level(s) deep
All within current directory
All within current site

Sleep interval between two consecutive queries :
[✓] For search engines : 5 to 10 seconds
[ ] For spidered URLs : 5 to 10 seconds

[ ] Autosize 'URL' and 'Page Title' columns in results grid
[✓] Sort results by 'Rank' upon querying the search engine

Base URL: - URL Title: -

Indexed Pages: - Link Popularity: - Alexa Rank: - Page Rank: -

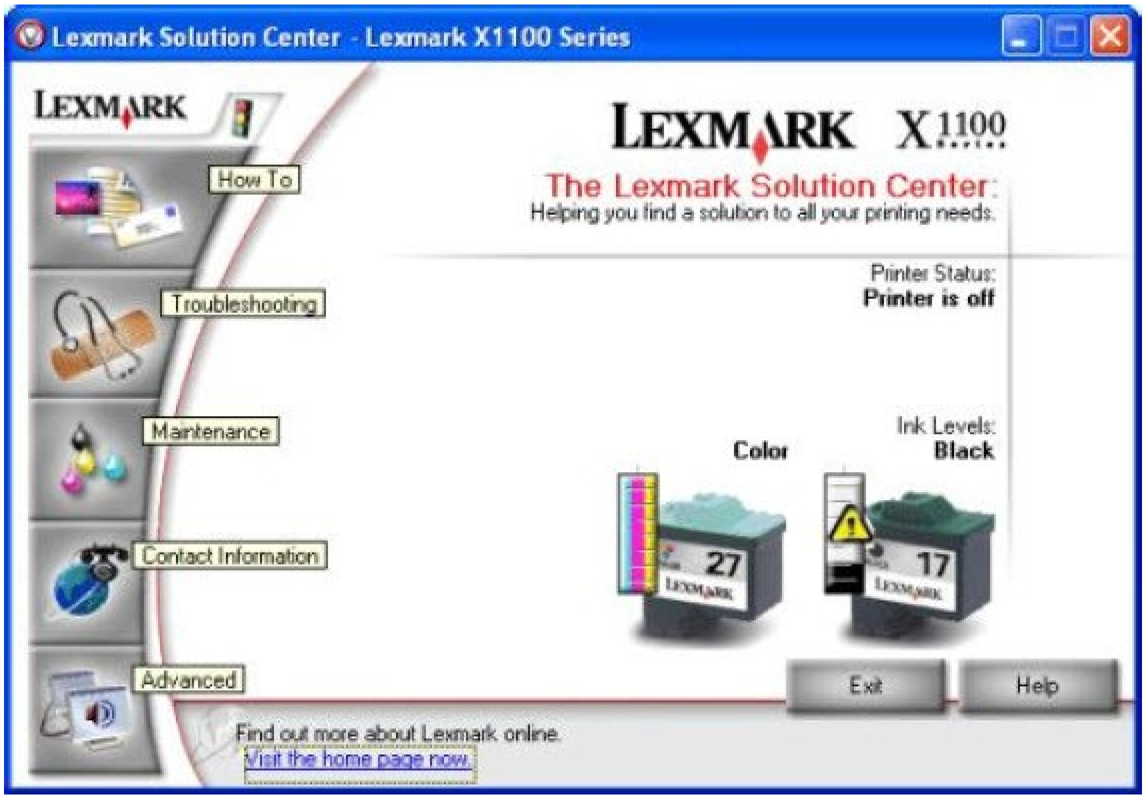
	URL

Keyword Prefix Filter : Keyword Postfix Filter :

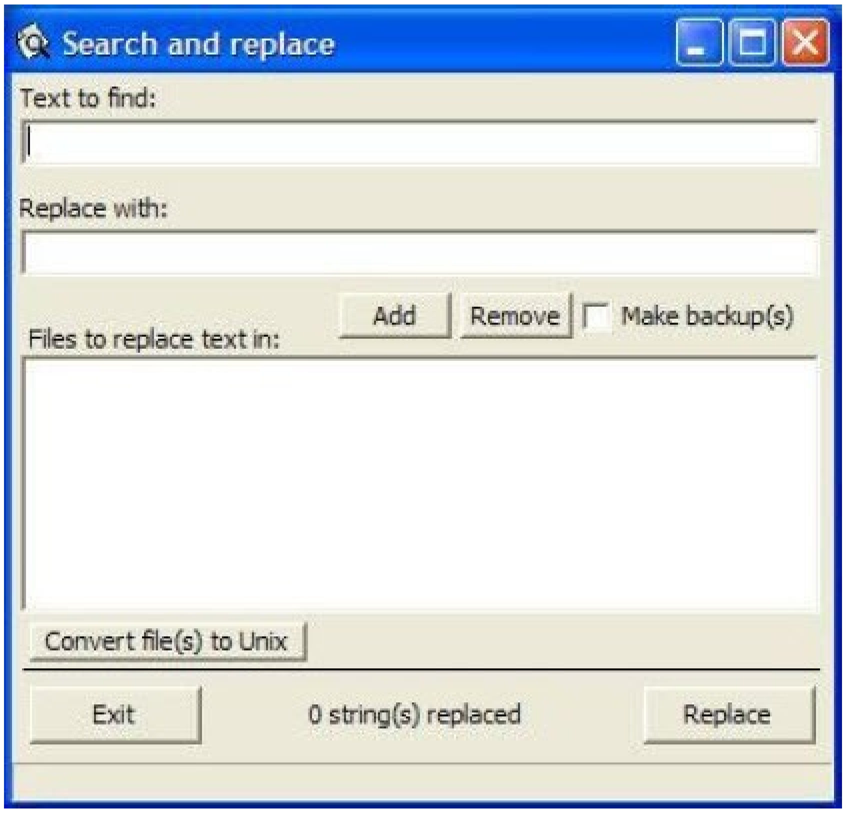
Total pages: 0 Current level: 0 Found on SE: 0

```
7/6/2006 4:57:11 PM: Application started...
```

Will it be an entry screen with navigation options...


Lexmark Solution Center - Lexmark X1100 Series
LEXMARK
How To
Troubleshooting
Maintenance
Contact Information
Advanced
LEXMARK X1100
The Lexmark Solution Center
Helping you find a solution to all your printing needs.
Printer Status:
Printer is off
Ink Levels:
Color
Black
Exit
Help
Find out more about Lexmark online.
Visit the home page now.

Or will it be something different...



Do you want extra buttons or links that the user can click in order to access your website or help documents? Will you even include help documents? And if you do, will they be built into the program?

Will you be distributing a free demo or limited-use version of your software? If you do decide to offer a limited-use version, will the user be able to access a registration link or payment button directly from within the program?

Will your software have periodic upgrades? If so, do you want the software to have the capability of upgrading automatically or will the user need to install an entirely new version?

What colors do you want? What type of font or images?

These are all things that need to be determined. If you can't decide on your own, you'll need to find someone who can assist you.

Depending on the software, you might also need to hire a graphic artist, someone who can professionally design things like custom buttons, icons, accessory images, enhancement graphics, and so on.

You also need to decide what the basic user interface will look like, both in size and appearance. As you can see from the examples shown here, the choices can be quite different. From very plain to very involved.

Naturally, a programmer will be able to make certain recommendations. After all, they're the ones with the expertise. But the more decisions you can make ahead of time, the better it will be for both of you.

Keep in mind, you're paying a programmer to create your product, which means you can certainly rely on them for advice about what can or can't be accomplished from a coding perspective. Just don't waste their valuable time on something as petty as the color scheme.

Programming Language

It's not crucial for you to understand or be knowledgeable about the different programming languages. However, it IS important that you know enough about the subject to make a qualified decision with regard to which one should be used for your own software product.

As I mentioned earlier, HotScripts is a great place to check out the different types of software products. But it also happens to be a good location for learning about the various programming languages and in what instances they're most commonly used.

My personal favorites – and ones I've used for many of my own software products – are PHP, Visual Basic, Delphi, and C/C++.

PHP

PHP (pre-hypertext processing) is a server side scripting language and is currently one of the most popular. It is often used to build dynamic websites and to connect with databases like MySQL.

An advantage of PHP is the fact that it's open source. What that means is that the software program can be run on virtually all computing platforms (Unix, Linux, Windows).

One of the reasons PHP is so popular is because it can be used to create software programs that are as involved and advanced as a community or portal system or as simple as a basic chatbox.

Programs I've created in PHP include Blogging Ninja, Synonym Miner, and Traffic Scorpion.

C/C++

C and C++ (pronounced C plus plus) are the most popular high-level programming languages and can run on both Windows and Macintosh systems.

Originally, C was used for writing UNIX programs but is now compatible with almost every available platform. C programs are known for their efficient use of memory.

C++ was built off the C language. Its object-oriented features makes programming easier and more efficient. And the reason most programs are written in this language is because of its power and flexibility.

Visual Basic

This is a type of object-oriented environment and programming language that was developed by Microsoft for the purpose of creating stand-alone Windows applications.

It allows programmers to quickly put together applications by writing code that accompanies onscreen objects such as buttons and windows. Software companies and developers most often use VB to build graphical client application interfaces.

Programs I've created in Visual Basic include Competiton Equalizer, Xyber Email Assistant, and Google Rank Analyzer.

Delphi

Delphi was originally a development tool for Windows applications, however, it currently works with both Microsoft.net and UNIX.

It has become popular because it’s as easy to use as Visual Basic but as powerful as C++. The language itself is an enhanced version of Pascal which supports object oriented programming.

Programs I’ve created in Delphi include SEO Spider, Site Searcher, and My Blog Announcer.

.NET

Although I didn’t mention this, I wanted to include .NET simply because it’s the one programming language I deliberately stay away from.

The primary reason is the fact that a lot of computers and servers don’t come equipped to run these types of program. In order to do so, another program would need to be installed.

Personally, I don’t feel it’s a good idea to create software in a language that requires people to go through extra steps just to use it.

Customer and Technical Support

The main difficulty with software is the fact that it will always require some level of customer or technical support.

No matter how much you prepare for, no matter how many questions and answers you publish or how extensive a knowledgebase or documentation you provide, there will always be someone who needs help in one form or another.

I know…

You’re probably thinking you won’t need anything heavy or elaborate until after your software business gets cranked up. You’re confident that you alone (at least, in the beginning) should be able to handle a few email inquiries.

Well, think again.

Even one product can result in dozens, if not hundreds, of weekly communication from customers, prospective customers, browsers, and people who have “general” questions or are simply curious about this, that, or the other thing.

And although many inquires might only require a brief yes or no, one or two sentences type of answer, many will not.

In most instances, you’ll need to provide qualified answers or information. That is, something much more involved than responses like, “Yes, we do provide free lifetime upgrades” or “for more information, please refer to our frequently asked questions”.

I don’t care how prepared you THINK you might be, there will always be questions and inquiries that were not anticipated or imagined. After all, you can’t possibly think of everything.

And believe me, when it comes to software products, there’s no possible way you can anticipate each and every conceivable question or inquiry.

But we’ll deal with that problem in a moment. Right now, let’s talk about basic methods you can provide in order to cut down the amount of work involved in customer and technical support.

First and foremost, you need to include adequate documentation with your software product. Although every instance would be different, there are three basic items that generally apply.

ReadMe file – This could include any number of things such as providing the user with a welcome or thank you message, a brief description of the product, how to proceed, installation instructions, terms of use, mention of any support forum or website, and contact information.

Installation instructions – Assuming you haven’t already included this information in the ReadMe file, you would provide clear and concise instructions that lead the user step by step through the entire installation process.

User instructions – Even if you have help files built into your software product, it’s always a good idea to provide independent instructions. That could be accomplished through an ebook or web pages, or both.

Naturally, you’ll want users to have access to all this information after they purchase but prior to actually installing the software.

The best way to handle this is to simply “bundle” everything in a zip file (the software product, instructions, etc.). That way, the user will have convenient access to everything related to the software. Plus, they’ll know exactly how to proceed.

The next thing you’ll want to provide is a set of frequently asked questions. For the most part, that should include three different areas of interest…

1. pre-sales questions
2. general questions
3. product questions

Try to come up with as many questions and answers as possible. A good way to do that is to simply navigate to other software product websites and view their FAQ's.

That should provide you with most of what you'll need.

And however you make these questions and answers available, make certain you have it set up in such a way that you can easily make changes. You see… No matter how many questions you initially come up with, others will always surface.

In fact, each time someone contacts you with a new or unique inquiry, you'll want to add that to your FAQ's.

The best method is to install a script that will allow you to manage all of your frequently asked questions on the fly. These are web based so you can log in to a private administration page and then make any additions or changes from there.

Aside from the convenience, this gives you the opportunity to have other people manage and contribute to your frequently asked questions as well. For example, someone on your support staff or even a friend or family member that helps out now and then.

Just go to HotScripts and type "frequently asked questions script" in their main search field. You'll find around a dozen scripts. And the good news is, some of the best ones are actually free.

Ok. That takes care of providing viewers with comprehensive questions and answers.

Next comes the knowledgebase, which is basically a more advanced and searchable version of frequently asked questions.

For most software products, this type of area isn't really necessary. But it's something you should keep in mind for the future, especially if you create and develop products that are rather involved or complicated such as paint/draw programs or website builders.

Depending on the type of product, you might also provide tutorials, either on your website or in downloadable ebooks.

For the ebooks, PDF is a good choice because you can simply create your tutorial in a Word document and then convert it to PDF. Plus, users can easily print out any or all of the pages.

A really good software product for creating ebooks is the one located at http://www.pdf995.com. It's totally free to download and use. However, in order to eliminate the ads that are displayed whenever you open the program, you'll need to pay a small "upgrade" fee.

Last but not least, there's the forum. This is where you set up a discussion board for your products and then users contribute to the information and content on an ongoing basis.

Although discussions boards are the perfect answer to self help, they also present distinct problems. The most glaring one is the fact that they don’t flourish on their own. Without some sort of guidance, the board will wither and die.

Also, boards almost always require one or more “authority” figures, people who can properly monitor posts as well as jump in and provide answers and information whenever necessary.

If you can eliminate the drawbacks, however, a forum can be the perfect solution to providing both customer and technical support. But that still might not be enough.

A more advanced method is to use a help desk system. That’s where users have the ability to submit tickets and then receive timely responses from a support staff.

One of the most advanced systems is Kayako (http://www.kayako.com). Among other things, it allows you to manage your support emails and you can easily chat with users in real time to resolve issues quickly. Plus, it also includes a knowledgebase, troubleshooter, and a download area for all necessary manuals, software, etc.

Naturally, you might not need anything quite that involved in the beginning, but like the knowledgebase, you should consider the possibilities, just in case. And again, HotScripts is a good place to conduct research and to see what other types of systems are available.

So now that we’ve pretty much covered every conceivable method of supplying information, there’s only one question left. Do you need to provide live customer and technical support?

That depends…

In spite of all the help documents, are you still bombarded with questions and inquiries? In spite of extensive instructions, do your users still need assistance in things like installation and operation?

It’s possible that your particular software products are going to require “hands on” assistance, something that can only be properly taken care of through phone conversation. If that’s the case, then you’ll probably need to set up a toll-free number.

In most instances, however, you’ll only need to provide the following items (at least, when you first begin)…

- basic help documents like ReadMe and installation instructions
- user manual or tutorial
- FAQ or searchable knowledgebase

Anything beyond that will most likely require the involvement and/or hiring of additional people. In other words, you won’t be able to handle it by yourself. But that’s not something you need to worry about right off the bat.

Just make certain you do two things…

1. Provide sufficient pre-sales information to put your customers at ease.
2. Provide sufficient post-sales information to fully satisfy your paying customers.

As long as you do that, your software business will be off to a solid and successful start.

To Partner Or Not To Partner

Although the idea of having a partner with regard to product ownership might not appeal to you (no one wants to share the profit), joining forces with someone can oftentimes fill a much-needed requirement.

For example…

You might need someone to help market your product. Or someone who is already set up to receive and handle customer and technical support.

Or, you might even need someone to put up the funds necessary in order to get your product created and developed.

Oftentimes, a person comes up with a terrific idea for a software product but doesn't have a clue how to take it beyond the initial concept. Or maybe they just don't feel confident about any of the tasks involved in creating and marketing software products.

That's fine. Because the most important ingredient in any software business success is having the ability to "imagine" the right products. All other aspects can be acquired through the act of paying for work or workers.

If you happen to be the "idea" person, if you're the one who can come up with terrific software products, AND manage to accomplish everything else, that's great.

On the other hand, if you fall short in any of the areas that are necessary in order to bring your idea from concept to financial reality, you might need to consider taking on a partner.

Just make certain that all the details of who does what, who is responsible for what, how much will be paid, and so forth, are carefully considered and documented.

And unless you happen to know the individual personally and intimately, you had better make everything 100% legal. That means drawing up a legitimate contract that both parties agree to and sign.

You might also need to decide whether or not you should create a partnership or form a corporation or LLC. That type of information is well beyond the scope of this publication, but it's definitely something you should look into.

The point, of course, is whether or not to even have a partner. And that's something only you can decide. Start by thoroughly evaluating everything you need in order to create and develop a sound software business.

Then, if it turns out you need (or want) a partner, choose as wisely and confidently as you can. And of course, draw up a legal and binding contract that will sufficiently and completely protect your interests.

Finding The Right Programmer

Unless you happen to know someone who can personally recommend a good programmer, you'll need to locate one on your own. Fortunately, there are convenient online services that can help you do just that. And most of them provide the means to assist you in doing it right.

Places like Rent A Coder, Elance, Scriptlance, and Guru, make it relatively easy to find and hire just the right programmer for each and every project you have in mind.

Rent A Coder
http://www.rentacoder.com

Elance
http://www.elance.com

Scriptlance
http://www.scriptlance.com

Guru
http://www.guru.com

In general, the steps for using these online services are as follows:

1. Post a project.
2. Receive bids.
3. Select a programmer.
4. Work one-on-one with the programmer.
5. Pay for the completed project.

We'll get into all the details and specifics about using my personal choice, Rent A Coder, further on. Right now, we just need to look at the overall process.

Mind you, this information isn't hard and fast for each and every service, but the following information will give you a general idea what to expect.

How It Works

The concept is pretty straightforward. You post a project on a public board and people who know how to do the job give you their bids.

Each of the people who bid on projects are registered with the online service. That means you have an opportunity to look through their profile and see what their qualifications are, how much experience they have, and what kind of work they've done in the past.

You'll also have an opportunity to see the total number of jobs they're performed while using that particular service, how much money they've earned there, and how these individuals are rated by the people they've worked for.

The rating of workers is similar to the feedback option that eBay provides. If someone is satisfied with the work that was performed for them, they'll say so. If they aren't, they'll say that as well.

Although a rating system could never be considered a one-hundred-percent accurate indicator, it can go a long way in letting you know how good or bad someone's track record is.

The point is, you won't be hiring someone blind. No, you might not know them personally, but there is generally enough information available at the website for you to make a qualified decision with regard to who you ultimately hire.

Of course, if you do hire someone who does a terrible job or doesn't fulfill their "contract" with you, you're under no obligation to pay them. That's another strong advantage of using these types of services.

When it comes to the handling of money, most of them use a form of escrow. Rather than money being paid directly to the programmer, funds are placed in an account (after a bid has been accepted). And those funds are only released to the programmer when the person who did the hiring is satisfied.

So it comes down to one very important detail… you don't pay until you get what you agreed to pay for.

Naturally, this type of system also protects the programmer. Someone can’t come in and hire them and then not pay when the job is completed and completed properly.

In other words, the programmer can’t be ripped off.

Using The Service

Although you can generally navigate these websites and look things over, you can’t really get into the inner workings until you register.

There’s no cost involved here, and for most services, it’s simply a matter of submitting a valid email address and choosing a username and password. Or, if the service uses your email address as your username (like Rent A Coder does), you only need to come up with a password.

Once you’ve opened an account, you can begin posting projects. But before you even think about that, you need to consider all the things that are involved. And the first thing on the agenda is how you’ll describe your project.

If your project is similar to a software product that’s already out there, you can use it as an example of what you like or don’t like. Or, to simply give the prospective programmer a better idea of what you have in mind.

One of the best ways to get some idea of how to present your project is to view others to see how they’re described. The more information you can supply, the more likely you’ll find the exact programmer you’re looking for.

However…

You need to be careful about what you post in this beginning stage. All you’re basically doing here is “trolling” for programmers.

When you first post a project, the only information you should reveal is what would be necessary in order for a programmer to determine whether or not the software product is something they would be qualified to do.

Yes, you need to be specific. And you need to be clear about what type of software product you have in mind. You just don’t want to give away so much information that someone else could “steal” your idea.

If you feel that you can’t submit ANY information without revealing too much, you can request that prospective programmers sign a non-disclosure agreement before they have access to your project information. (Rent A Coder allows you to choose this option on the first page of your project form.)

You need to understand, however, that doing this will limit the number of bids you’ll receive. So only use this option in the event you have no other choice.

That being said, here are all the things you’ll want to include in your project description.

- detailed description of what you need/want
- required programming skills or expertise

- any budget or timeline constraints
- turnaround time for drafts and revisions
- brief description of you and/or your business
- your level of knowledge with regard to the project

By supplying this type of information, you not only make it clear what you need, you'll be more likely to attract the right kind of programmers with the right expertise.

Of course, nothing's written in stone. My own project posts don't include near that amount of information. For example, here's one I posted for a desktop application…

This is an easy project for a coder who knows what they are doing...

I need PC software to merge .txt articles into one master .txt file.

A list of features will be provided upon request. DO NOT bid unless you can finish this quickly and can begin work right away. I have more work for the right coder.

Please send over any questions or comments. Thank you!

As you can see, I focus more on getting someone who is qualified to carry out the terms I'm looking for. And although I include enough information to indicate the work I need, I reserve details of the project for future contact.

Again, take a look at other project descriptions. In fact, before you submit your own project, you should spend a fair amount of time just looking things over, getting familiar with the process and how everything works.

Choosing A Programmer

Once you've posted your project and received bids, you'll need to decide which one is best for your particular needs.

Of course, if you don't like any of the choices, you don't have to consider (much less accept) any of the bids that are presented.

Assuming that's not the case, it's simply a matter of checking their profiles and then communicating with the most promising programmers, ones you feel would be well suited for your project.

Just remember that, above all else, you need to be able to work comfortably with the person who will be creating your software product. If you can't do that, it's very unlikely that your product will turn out the way you want. And even if it does, the process of getting there could have been unnecessarily stressful.

Naturally, you want someone who is highly qualified. But you also want someone you can freely talk to, someone you can easily communicate with throughout the entire project. All in all, you need to feel confident that you're both on the same wavelength.

The projects you post will be seen by programmers world wide. And the advantage is the fact that various individuals and companies will be actively bidding against one another in order to get the job.

The disadvantage is, various individuals and companies will be actively bidding against one another in order to get the job.

It's an advantage because it affords you the possibility of getting the work done for an extremely reasonable price. It's a disadvantage because there's the temptation to hire someone based on the cost of their services.

Whatever you do, don't fall into that particular trap. The ONLY thing you should be concerned about is the quality and expertise of the programmer. If you can hire one for a reasonable (or unbelievably low) fee, even better.

That's why these online services are so great. They give you the opportunity to have programmers compete for your work. Just don't allow yourself to be tempted simply because the price seemed too good to pass up. Remember… If it sounds too good to be true, it probably is.

Pay close attention to the profiles, ask questions, thoroughly review their past jobs.

In other words, do your homework.

The more attention you pay to the person or persons you're considering for the project, the greater the odds that everything will go smoothly and professionally.

Paying For Your Project

Once you choose a programmer, you'll need to place whatever money has been agreed upon into an escrow account. This is handled by each of the online services and makes paying a programmer much easier and safer.

Generally, funding can be done with a check, through credit card, or wire transfer (bank to bank). And with some services, like Rent A Coder, you can also transfer funds using PayPal.

With regard to when you actually pay the programmer, there are two choices. You can pay a lump-sum amount at the end of the project (my personal preference), or you can establish what are referred to as milestones.

These would involve partial payments that take place at specific intervals throughout the project, ones that have been agreed upon by both parties. For example, you could pay twice. Once when the project reached a halfway point and the balance when the project was completed.

How many milestones you specify is up to you and the programmer. It also depends on the project itself. Most often, milestones are only used in more complicated and longer-range situations (where the project will spread out over an extended period of time).

For anything simple that will take a relatively short period time, however, one amount is generally paid at the end of the project.

Of course, the programmer doesn't receive any money until you say so. When the work is completed to your specification (at a milestone or at the end), you verify that fact and then authorize the release of funds.

If there's a dispute, the service will appoint an arbitrator to resolve the issue. This individual looks at both sides – exactly what you asked for and what the programmer delivered (or didn't deliver) –

and then makes a determination with regard to the completion of the project.

Since both parties have agreed to abide by the arbitrator’s decision going into this process, whatever the arbitrator determines is binding.

In the event a programmer is unable or unwilling to satisfy the original agreement and complete the project, you can simply choose another programmer.

Using Rent A Coder

Rent A Coder
http://www.rentacoder.com

This one is my personal preference, the service that I use exclusively to have software products created and developed.

Although I wouldn’t necessary say there’s anything wrong with any of the other services, I would have to say that Rent A Coder is far superior when it comes to hiring programmers. At least, that’s been my own experience.

Opening an account is free and simply requires having a valid email address. You’ll also need to verify your email address with a temporary password that will be sent to you. Once you’ve done that, you can choose a permanent password (which gives you unrestricted access to the entire website).

When you complete the signup, you’ll be taken to the first page of the form for submitting a project. Unless you’ve already worked out all the details ahead of time, I suggest that you skip this and browse through the entire Rent A Coder website first.

A good place to start is the frequently asked questions (actually, it’s more like a series of help documents in the form of questions and answers). Once you’re inside the main area of the website, just click the “Buyer FAQ” link on the left-hand navigation menu.

Beyond that, here are some suggestions for achieving the best results (many of these would apply to all online services, not just Rent A Coder).

<u>Quick Tips and Advice</u>

Before selecting a coder, always find out how long after the completion of the project they will provide support and/or bug fixes. Make certain the time allotted is at least six months.

Review the number of Mediations/Arbitrations the coder has had and how many they have lost. If the number is high, you should definitely pass them over.

Look at the number of Jobs In Progress. Are they already working on too many projects? This could be a sign that they might not be able to complete yours on time.

Have they had any Missed Status Reports? Too many of these would indicate that the coder does not conduct themselves in a professional manner.

What is their area of expertise? Study this carefully to see if they have experience with the technologies required for your job.

I personally don’t choose coders unless they have a rating of 9 or higher. This works very well for me. However, it is still possible to choose a coder with an excellent rating and then have problems.

Most of the coders who bid on your projects will be based outside the United States.

How good is their command of the English language? Is it easy or difficult to communicate with them?

You could mention in the description that you will pay a bonus if the job is completed on time.

Thoroughly beta test each version of the software that the coder gives you. No matter how good they are, you will always find bugs. It is also a good idea to have at least one other person beta test the software before signing off on the project.

I normally like to complete a project with the coder in 14 days or less unless it is a small project. If that’s the case, I will allow 3 days for completion. Use your best judgment when deciding upon the timeframe. Some coders will take advantage of you if you let them.

Do not select a coder simply by their rating and the number of jobs that they have completed. ALWAYS review their past jobs to find out if they have ever done anything similar to your project.

Also, pay attention to the average price they have been paid per job. Every time I post a project I always have coders with excellent ratings placing bids. However, many of these coders have only worked on projects where the average price paid was less than $50. Most of my projects cost much more than this, so I would conclude that they do not have the experience needed to do the project correctly. It would be too advanced for them.

On an open auction, certain information is made public after a coder has been chosen. If you would like to keep the details of your bid request private, you should select “Auto-privatize my open-auction bid requests” (under My Alerts / Other). That way, once you’ve selected a coder, all open auctions will automatically switch to the more protected private auction.

Project Example

Following is a project I posted on Rent A Coder along with some of the actual dialog.

First, the project listing…

This is an easy project for a coder who knows what they are doing...

I need software that allows me to replace text within an HTML document. I should be able to batch process HTML files. All instances of text will be replaced and HTML files automatically resaved.

DO NOT bid unless you can finish this quickly and can begin work right away.

I have more work for the right coder.

Please send over any questions or comments. Thank you! You’ll note that whenever I post a project, I always include the same wording at the beginning and the end. The only thing different is the project description.

I received numerous responses to my post, however, this one immediately caught my attention…

I already have this 75% completed, would take about 30 minutes to finish.

Larry

This was my response to Larry…

Hi Larry,

Great! I want the ability to choose more than one file at a time for processing.

Jeff

Once we agreed on price and conditions…

Project phase change: All funds have been escrowed. Waiting for work to be completed by Larry_D (the coder).

Following is some of our continuing communication...

Jeff:

Quick question... I accidentally converted a CGI file to DOS. How do I convert it back?

Larry:

It’s a matter of replacing the “new line” signal characters with UNIX compatible ones.

I’ll be posting your application here shortly.

Jeff:

Can you add this functionality to the software?

Larry:

Sure thing Jeff.

Jeff:

Thanks for adding that! It’s a CGI file, but the software doesn’t give me the option to select a CGI file.

Larry:

I have Added CGI and *.*

Jeff:

Thank you Larry! I’ll sign off now and rate you. Be sure to rate me too. :)

Since everything had been completed to my satisfaction, I signed off on the project which in turn released payment to the programmer…

Jeff Alderson (the buyer) accepted 100% of work and released $10.00 from escrow to Larry_D (the coder).

Naturally, that’s a very simplistic example. Most software projects will be a bit more involved, both in selecting a programmer and the dialog that takes place through the project.

But whether it’s something short and simple or something relatively long-range and complicated, the process is still the same…

You post a clear and accurate description of what you want, you choose from the programmers who bid on your project, you exchange dialog with that programmer, and you receive your software product.

The only thing left to do after you sign off on the project and release funds is to rate the programmer. Just keep in mind that the programmer has the ability to rate you as well. In other words, it goes both ways.

No matter what happens throughout a project (or even beforehand), always conduct yourself in a professional manner. That way, you’ll retain a good solid reputation, one that programmers will appreciate and respond to.

Your Product Website

When it comes to software products, there are two types of websites that are most commonly utilized.

Sales Letter – This is basically a single page that is devoted solely to talking a prospective buyer into purchasing one specific product. Aside from the order link, the only other navigation would generally lead to secondary pages such as contact information, privacy policy, terms for using the site, and affiliate signups.

Product Website – This would be a fully involved “umbrella” site that highlights and promotes all software products. Although it would most likely have individual sales pages for each product, it would also include a variety of help and information pages that contain things like FAQ, knowledgebase, tutorials, forum, and so forth.

One of these websites is not necessarily exclusive of the other. Just because you have individual sales page websites doesn’t mean you won’t develop an umbrella site as well.

Either way, here are some of the more important components, starting with your basic sales letter.

Sales Copy

As previously mentioned, software doesn’t require near the kind of “hard sell” effort that other products do (for example, ebooks).

Mind you, that doesn’t mean you can sit back and relax and your software will sell itself. You still need good “killer” sales copy. You just don’t need to spend quite as much time convincing your buyers why they should purchase your product.

What you do need to concentrate on, however, is all the features and benefits your software provides. Above all else, that’s what people want to hear. And although benefits are generally considered more compelling than features, don’t get too carried away.

When people are looking to buy software, they want to know what it can do for them.

Does it automate something? Does it make a particular task easier? Will it help them drive more traffic to their websites? Can it provide them with hours of entertainment? That’s what they want to know.

Beyond that, you need to be very specific about all the features your software product contains. That would basically include almost everything on that list you originally created, the one that would define the characteristics of your product.

Go to the sales sites of several other (hopefully, similar) software products. Take a look at how they talk about the features and what type of features they include. Did you come away knowing exactly what the software could do and what features it had? Or did you still have questions or doubts?

Once you’ve seen how the sale of other products are handled, you’ll have a much better idea what will and won’t work. Then simply take the best aspects and incorporate them into your own website.

<u>System Requirements</u>

One of the most important aspects of selling software is letting prospective buyers know where and how they can use your product.

Make certain you’re absolutely clear about what is required in order to install and/or run your software. For example, compatible operating system, the load requirements (memory, server, or hard drive space), Perl module, MySQL, cron capability, and so forth.

The last thing you want is to have people purchasing software that won’t work with their particular operating system or can’t be run on their server. All that equates to is unnecessary refunds.

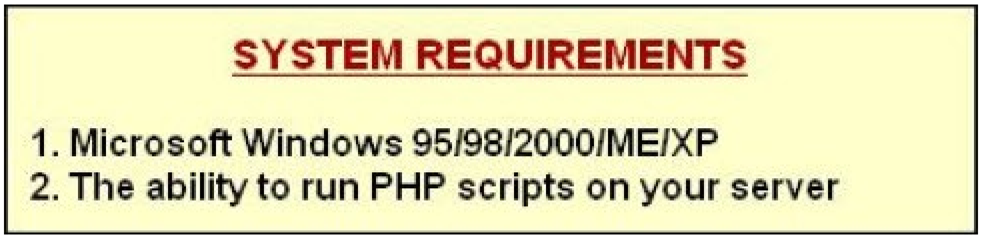
So be sure you place the information in such a way that viewers can readily see it (close to the order button is always a good choice).

For my own products, I either include a systems requirement image like these…

REQUIRED
PHP 4.0+
MySQL

Designed for
Microsoft®
Windows® XP,
2000, ME, 98

Or, I simply create a highly visible text area…



SYSTEM REQUIREMENTS

1. Microsoft Windows 95/98/2000/ME/XP
2. The ability to run PHP scripts on your server

It doesn’t really matter how you include the information. Just make certain any and all system, installation, or operating requirements for your software product are clearly and accurately stated.

Frequently Asked Questions

Your sales copy is meant to sell your product. But regardless of how thorough your sales content might be, there will always be questions that prospective buyers have.

By providing a list of frequently asked questions (and their answers) you can cut down on the amount of inquires your receive. And more importantly, you can help reinforce a prospective buyer’s confidence about you and your product.

You also have to consider the fact that many people prefer not to read an entire page of sales copy, no matter how compelling or exciting it might be. They prefer to get right to the facts. That means heading straight to the FAQ page. If you don’t have one, you could risk losing a large number of potential buyers.

The number of questions you include can vary considerably from product to product.

If you’re not sure about how many or even what type of questions and answers to provide, just look at some FAQ pages on other software product websites.

Installation Instructions

Although you’ll be including this information with your software, it’s always a good idea to provide the same instructions on your website. That way, they can be easily accessed at any time.

Also, for some prospective buyers, the instructions can be the deciding factor in whether or not they purchase your product. And just like supplying people with the installation and operation information, this can help prevent the number of refund requests you receive.

Product Manuals

This falls into the same category as the installation instructions. Although you’ll be including it with the software product, you might also want to make it available from your website, as HTML pages or a downloadable ebook.

Naturally, you won’t want to include a manual if it’s going to give away any kind of trade secrets. Or lead someone else to develop a piece of software just like it.

The most common manuals would be those involved with programs like paint/draw or similar products. That way, people can get an idea of all the types of things they can do with your software prior to purchasing.

But again, it all depends on what type of software we're talking about. Just use your own judgment. If you think posting the manual will be to your advantage, then go ahead and make it publicly available. If not, then simply restrict distribution to actual buyers.

Tutorials

Naturally, some products won't require any form of tutorial. The instructions that come with them will be more than sufficient. However…

Let's say you have a product that automatically submits articles to various online directories. In order to help people get the most use out of the software, you could publish articles and/or tutorials about various aspects of writing and distributing articles.

For example…

- how to write compelling and interesting articles
- how to find niche topics to write about
- how to optimize your articles for specific keywords
- how to locate and compile good keyword lists
- how to write good descriptions of your articles

Basically, you want to provide any and all information about topics that would directly or indirectly relate to using your software product. If you place this type of information in the form of an article, you yourself have the ability to gain benefit through online directories.

And of course, every page that gets listed in the search engines gives you that much better opportunity to receive additional targeted viewers. Or should I say, prospective buyers. Definitely a win-win-win situation.

Contact Information

How much contact information you provide is entirely up to you. The most common contact methods are email address, phone number, and mailing address.

Depending on the volume of inquires you receive, how much or how many contact methods you provide is something only you can decide. When it comes to online methods, however, you have two basic choices.

Naturally, you can provide an email address so that people can simply send you a direct message. If you don't have too many inquiries and you can handle them quickly and easily, that might be all you need.

A better choice, though, is to provide a form for people to fill out. That gives you the opportunity to separate messages by subject categories and priority. A big help in knowing what messages you need to answer first.

The Purchase Process

Although there are many methods you can use to allow people to purchase your software products online, there is only one I would personally consider for digital products (those that can be delivered through download).

The service is ClickBank (http://www.clickbank.com). And there are several reasons I chose them.

For one thing, you only need to set up one account (for a one-time fee of $49). That gives you the opportunity to sell as many products as you like.

Secondly, they have a built-in affiliate program. That means you can quickly and easily initiate your own affiliate network, allowing other people to earn a sales commission for selling your products.

Third, ClickBank offers an extensive marketplace that is well organized by specific categories (http://www.clickbank.com/marketplace/). In that area, they list over 10,000 products that are ranked by popularity.

This provides a convenience place where people can quickly and easily locate your products. To either purchase them, sell them as an affiliate, or both.

And since ClickBank is so popular and widely advertised, you can count on receiving considerable promotion and marketing benefit. So the only thing you really need to do is provide a good quality product.

You need to be aware, however, that ClickBank has a money-back guarantee policy (60 days as of this publication date). If a customer is dissatisfied and wants their money back, ClickBank will readily honor that request. In general, with no questions asked.

That’s fine if there’s a legitimate complaint. Like they never received the product. Or the product can’t be installed or doesn’t work on their particular operating system (yes, it’s clearly stated on your website, but that’s not the point).

The problem is, ClickBank’s refund policy leaves your products vulnerable to people who want something for free. They know that once they’ve purchased and downloaded a product, they can immediately turn around and request a refund. And if you don’t give it to them, ClickBank will.

There are two ways to combat that type of situation…

1. You can use a different purchasing system, one that doesn’t have an automatic refund policy. For example, if there’s a dispute, PayPal’s only concern is whether or not the goods were delivered, not the quality of the product or customer satisfaction.

2. You can incorporate a utility or system that will disable your product from a remote location. That way, if someone asks for and ultimately receives a refund, you can simply “turn off” their product, preventing it’s use. Think of it the same way a trial or limited-use version is configured. Once the time is up, the software shuts down.

Mind you, neither of these choices is perfect. What you gain with regard to some other merchant provider’s refund policy (or lack of one), you will most likely sacrifice in terms of ClickBank’s other benefits, like the affiliate network and the marketplace.

And having a programmer create a system for remotely disabling software means more expense (maybe a LOT more expense). But of course, it could be well worth it when you consider how much money can be lost to bogus refunds over an extended period of time.

Of course, the ultimate choice would be no money-back guarantee, no refunds – aside from very extreme or extenuating circumstances. Even then, it would be your call and not an arbitrary decision by a company who has absolutely nothing to lose by giving someone their money back.

After all, you don’t see Microsoft giving back money after you’ve purchased and downloaded one of their products. Or any other software company. Try asking for your money back with one of Adobe’s (http://www.adobe.com) products and see how far it gets you.

The bottom line is, you can make a tremendous amount of money selling software products. And sometimes, you have to simply bite the bullet and accept things for what they are.

ClickBank offers a great system for promoting and selling products. The fact that they offer what seems at times to be an unreasonable refund policy (at least, for sellers) just happens to be the only fly in the ointment. However… If you can embrace that particular drawback and turn your attention to all those aspects of your business that you do have total control over, you’ll be much farther ahead.

Promoting Your Products

This is where it gets really intense. After all, if no one knows about your product, they can't very well purchase it can they?

As previously mentioned, the ClickBank marketplace affords you a great opportunity to get your products noticed. That's why it's important that you initiate your own affiliate network right away.

The only difficulty is the fact that so many other ClickBank product owners are also vying for affiliates and increased sales. Unless you have a unique product that no one else has, you'll need to be prepared to compete on whatever level it takes to get your fair share. Not impossible, mind you. You'll just have to work harder at getting the affiliates attention.

In order to induce a receptive response and ensure that enough ClickBank people (affiliates) will successfully promote your product, you'll need several key ingredients.

- a quality product
- a professional sales page with high conversion capability   a generous 50 or 75 percent sales commission
- a very compelling product description

As long as you supply those four things – and the level of quality in both the product and sales page are relatively high – you should be able to enjoy a substantial monthly income flow.

Be aware, however, that affiliates are only as good as the sales materials they're using. And not every affiliate is capable of coming up with their own killer ads and sales content.

The more help you can give them, the higher the financial results. Give them good sales copy, high-conversion ads, quality images, and solid marketing and promotion tips and information. That's how you wind up with a winning and productive sales force.

Ok. Getting back to your own promotion…

Some of your most effective marketing and promotion will originate from “script” websites like SimplyTheBest (http://www.simplythebest.net) and HotScripts.

And, if you happen to offer a trial version of your product, you should definitely take advantage of download sites such as CNET (http://www.download.com) and Upload (http://www.upload.com).

Millions of searches are conducted on script and download websites every week, which means the chance of someone finding your software product will be substantially increased.

When viewers are looking for software products, they conduct searches. And those searches are generally focused on two different areas. One, the actual search engines like Google and Yahoo, and two, all the major (and even minor) download and script locations.

Of course, since many of the pages from download and script locations are also listed in the search engines, viewers oftentimes find what they’re looking for without having to go any farther than Google or Yahoo.

But you can’t rely entirely on having the pages where your product is listed showing up in search engines. Or being displayed in high enough results positions that someone would even learn about your product.

You can, however, count on your product being located by the people who search download and script locations directly. So the first thing you need to do is get your product listed on as many of those types of websites as possible. And don’t just settle for the major ones. Look for specialty and lesser known services as well.

Like anything else, the more places you can get exposure, the better. And getting your product on the download sites gives you a good solid start. But once you’ve submitted to those locations, you need to heavily concentrate on your own search engine tactics.

Since your sales pages is focused on selling your product and not necessarily optimized for certain keywords, it probably won’t do you much good in terms of search engine benefit.

Of course, if your sales page happens to also score well for keywords, all the better.

Just don’t sacrifice the power of your sales content merely to achieve higher keyword optimization.

You’re much better off creating independent pages that are keyword specific. In other words, create promotion pages whose content is developed around terms and phrases that are being actively searched for by prospective buyers.

And the best way to present the content of these additional pages is in the form of articles. That way, you can submit them to article directories and receive even more promotional benefit.

Last but certainly not least, you need to seriously consider paid advertising. The best method by far is to implement Google Adwords campaigns. That’s one of the few techniques which can give you the ability to accurately reach your target audience or audiences.

Whether or not you run campaigns for an extended period of time, Google Adwords is an excellent means of “jump starting” your marketing and promotion. And that’s exactly what you need at the very beginning.

In Closing

There’s absolutely no doubt whatsoever that vast amounts of money can be made in the software industry. And creating and selling your own software is one of the best methods.

More importantly, it’s one of the easiest methods. That’s assuming, of course, that you’ve allowed yourself to become totally and completely familiar with all aspects of this type of business.

Like any other business, the more you know, the more confident you’ll be. And the more confident you are, the more likely you’ll reap considerable financial reward.

Beyond that…

The only thing that could possibly hold you back is not taking action. Don’t just sit there thinking about all the software products you can create, or how much money can be generated once those products are on the market.

Take all the resources you’ve gained here and put them to work for you. Not tomorrow, not next week, but right away. I guarantee…

Once you get started in this business, you’ll never want to stop!