Software Experts

Sample

Statement of Work

Produced for

GREAT SOFTWARE

18813 Tracer Drive
Lutz, FL 33549

www.swxperts.com
ceo@swxperts.com

Statement of Work

Date: February 5, 2007

This document is the Statement of Work between Great Software, Inc. (the "Company,") and Software Experts.

Company Name:	**Great Software, Inc.**
Service Provider Company/Name:	**Software Experts**
Project Name:	**Project SG**

Description of work to be performed.
Software Experts shall provide services in developing the application as per the specification below. Customer shall provide all the information requested by Software Experts. Failure to provide it in a timely manner will cause delays in the project.

1.1 Status

The description below describes software requirements for the Ignoble form design application. It is a normative appendix for a contract between software vendor and customer. However, it does not have any meaning independent from the software specifications provided by the Customer. The list of original user specifications is provided in the Appendix A.
In areas described herein and Customer's specification, this very description takes precedence. Any changes conflicting with the original specifications (including any requirements changes, additional functionality and any other changes), this document are subject to start the Request For Change process.
Also the document describes proposed technical solution for the implementation of the product.

1.2 Should/Will/Shall words meaning

The terms "must," "should," and "may" imply the following:

1.2.1. Must/Shall/Will

This is an absolute requirement for delivery in this release.

1.2.2 Should

Inclusion in this release is dependent of factors such as time, cost, and complexity of implementation.

1.3 Business Requirements

The main is non-technical people wanting to quickly build a simple web-site with form. So the application must hide all complexities of such operations.

1.3.1 Installation and functioning should not require administrative privileges.

1.3.2 The application should hide technical details even by price of decreased functionality.

1.3.3 The application should provide a comprehensive user manual and support environment.

1.3.4 User interface must be as simple as possible.

1.3.5 The application architecture must be flexible to include additional functionality in subsequent releases.

1.4 Software Requirements

1.4.1 Application framework

The application will be built on top of Eclipse RCP framework to provide cross-platform support. Main menu and toolbars will be implemented as RCP Workbench contributions.

1.4.1.1 Extensibility

The application will be designed and implemented to allow easy extension of functionality of the result product. In order to achieve flexibility and extensibility modular design will be used. This will allow us to develop and maintain the application with low efforts.

1.4.2 Form Design

1.4.2.1 Blocks

The main building block of a Ignoble form will be block as in terms of CSS Level 2 layout. The form will be a collection of blocks without possibility to embed blocks into each others.

1.4.2.1.1 Positioning

The form will be designed in a WYSIWYG manner. User will select a form component and place it onto a Canvas. The system will use absolute positioning to layout components on the users form. In other words form will be a collection of blocks with CSS absolute positioning.

1.4.2.1.2 Appearance

The application will provide ability to change border style of each block. This includes width, color and style of each border segment (top, left, bottom, right).
The application will provide a way to change background color of each block.

1.4.2.2 Inline layout

HTML inline flow will be used to layout text inside text blocks (see below).

1.4.2.2.1Font styles

The application will allow change font style of a piece of the text. This includes size, family, weight and decorations.

1.4.2.2.2 Spell checking

The form designer will check spelling of the text in the text blocks and labels and draw red line under misspelled words.

1.4.2.3 Form handling

Each form will be stored in the database as a table.

1.4.2.3.1 Versioning

Each database used by Ignoble form handling module will contain a table named \`Ignoble_form\`. The table will store versions of the Ignoble form tables.
The form handling module will check if the database table has valid version before try to write data into it. In the subsequent releases of the application a data migration module should be developed to allow user upgrade forms transparently.

1.4.2.3.2 Fields

In order to gather information a notion of field will be used by the application. Field is an editable piece of information stored in the database.
Each form field will be reflected onto one column of the database field directly.

1.4.2.3.2.1 Domains

Each field can be associated with one of data domains:

- String
- Text
- Number
- Phone
- E-mail
- Date

The field value will be cast to appropriate data type before write into the database.

1.4.2.3.2.1 Field types

The full list of the field types presents in the original Ignoble application specification.

1.4.2.4 CSS support

The visual editor will support CSS class assignments for block elements in the volume enough for customization of block appearance in the editor. However, no effort will be maid to account style inheritance and CSS Level 2 features.

1.4.2.5 Clipart

Insertion of raster image files in the following formats will be supported via image form component:

- PNG
- GIF
- JPEG
- ICO
- TIFF
- BMP

The images should be inserted as is, without conversion or editing.

1.4.2.6 Auto-shapes support

The application will provide support for auto-shapes by rasterizing them during code generation phase. So user will be able add following figures onto the form:

- Lines with/without arrows on the ends
- Rectangles
- Ellipses
- Rounded rectangles

1.4.3 Form storage

1.4.3.1 XML Format

The form data will be stored in XML format with XStream library. The library produces light-weighted XML representation of a model graph with nearly zero effort.

1.4.3.2 Versioning

The XML will use processing instruction construct to distinguish different versions of the form data. For example the following instruction can be used at the beginning of the XML file:
<?xml version='1.0' encoding='UTF-8'?>
<?seagrean version='1.0'?>

1.4.4 Code generation

1.4.4.1 Target environment

The first release of the form build will be targeted to the LAMP (Linux Apache MySQL PHP 4) platform.

1.4.4.2 Generation results

Ignoble code generator will produce following files:

1.4.4.2.1 Form meta-information

A file with meta-information describing form fields, their type, and database connection parameters.

1.4.4.2.2 HTML

Is a plain XHTML document referencing all other form related files. The style information will be applied via linked CSS with identifier selectors if block classes was not specified in Visual editor.
The HTML file will not contain any server-side script to ease editing of the form.

1.4.4.2.2.1 Validation

The result HTML also will exploit javascript to valid field contents. In case of invalid values form submit fill be abandoned and an alert window will be shown with error description.

1.4.4.2.3 CSS

Is a stylesheet with style information for the form. Due the automatic management style class names will be autogenerated.

1.4.4.2.4 Images

Rasterized auto-shapes and images inserted into the form by users.

1.4.4.2.5 Server-side form handling modules

Actually code generator will copy predefined versions of the server-side modules wrote by Vendor developers to handle form submit.

1.4.5 Code editing

The application will allow edit code generated for the form implementation. The software should try to reverse-engineer modified HT ML code into form model to build correct representation in the visual editor. However due nature of visual designer this is not always possible.

1.4.5.1 HTML/Code editor

The editor will be based on Amateras HTML editor (CPL 1.0). All changes maid into the original code will be pushed back into the main source tree.

1.4.5.1.1 Structure highlighting

The editor will support HTML structure highlighting.

1.4.5.1.2 Tag auto-completion

The code editor will provide hint for tag auto-completion.

1.4.5.1.3 Printing

The editor must provide ability to print form code with structure highlighting. Standard Printer/Page setup will be used to adjust printing parameters. The page will be printed as it rendered on the screen. No additional elements/styles will be introduced.

1.4.6 Administration setup

1.4.7 Form import

Ignoble application will provide ability to import forms from other file types. The application should try to obtain meaningful labels for imported fields if possible. The layout of original form will be abandoned.

1.4.7.1 Supported formats

1.4.7.1.1 PDF

Only the latest PDF versions support forms. So import of these PDF documents will be supported.

1.4.7.1.2 HTML/PHP/ASP

During import of an html form, all php & asp code will be ignored.

1.4.8 Form merging

The application will provide a form merging wizard that will allow insert form into a convenient HTML page template. This will be done by loading the template into embedded browser control with interception of all HTML events by injected javascript. The javascript will insert a colored block to the form template to mark position of the form insertion. As Ignoble form is a block from CSS 2 layout view, the script will insert a block element (div) also. This will allow imagine how the result of the merge operation will look like.

1.4.9 Form upload

1.4.9.1 Catalogs structure

During upload a root folder for upload will be specified. Directory structure description below is relative to the upload root.

Folder	Description
Ignoble/<version>	Storage for the Ignoble specific code. <version> is a version identifier of Ignoble server-side form handler.
<form name>	The name of the html file for the form handling
_Ignoble_forms_/<form name>/<form version>/	Directory storing form specific files such as images, javascripts, css and so on.

1.4.9.2 Protocols

The application will allow upload of the form via following protocols:

- FTP (Active or passive)
- SFTP (FTP over SSH channel)

Later another transfer protocols can be added.

1.4.9.3 Firewall & Proxy support

The application should try workaround protocol constraints by switching protocols modes (Active/Passive) and protocols in case of connection failure.
However, the only way to pierce some corporate firewalls is usage of proxies. The application should support HTTP/FTP/HTTPS proxy if appropriate for the protocol.

1.4.9.4 Upload verification

At the end of upload procedure the application should verify validity of the remote code copy. It will be done by download of the form data and comparing with the original local copy.

1.4.10 Server-side modules

Server-side modules will be reused by all site forms generated by Ignoble. So form meta-information data will be used to handle form submits correctly.

1.4.10.1 Form handling module

So form meta-information will be used to obtain form handling information.

1.4.10.1.1 Validation

The module must check validness of the submitted form data before write into the database. In case of invalid data, the module should show error page explaining cause of the problem.

1.4.10.1.2 Database schema creation

At start the module should connect with the database and check presence of the table used to store form data. If the table does not exist the module should create one.

1.4.11 Update manager

1.4.11.1 Automatic update

The update manager shall allow automatically download and install updates and security fixes. However major updates and upgrade from Free to Pro versions should be paid on the e-commerce web-site. So the manager will track compatibilities between major version numbers and offer to update for the same major release only. Availability of updates will be checked on the first run of the application and with 7 days interval after then. This will allow to lower load on update servers after release of a new application version.

1.4.11.2 Paid functionality access

This will be achieved by distribution of the same binary codes for both Free and Paid application releases. Paid functions will be opened on the correct activation of the product.

1.4.11.3 Update manager extension

Special extension of the Eclipse update manager will be used to check whether the public update can be installed for free. To check availability of the public update, the update manager will check a file on the update server named updates.xml.
The file will contain information about updates available for the all application versions. This file will also be used to track serial key compatibility between application versions.

1.4.12 Serial numbers/licensing

Serial number is a sequence of alpha-numeric character uniquely identifying sold copy of a media with the product. A certificate with serial number will be distributed with software boxes or will be provided via e-commerce web-site.
A freshly installed application will function in the Free mode. To get access to Pro functionality user must activate its application copy by connecting with Ignoble license server and providing serial number. In case of successful activation, Pro functionality will be unlocked until next reinstallation of the software.

1.4.12.1 License server

The initial version of the license server will not provide any administrative interface or piracy prevention. The server will accept all license requests for existing product

number by default. However detailed logging of the operation will be used to incident tracking. In the subsequent releases if needed we can add constraints onto number of installations per serial number and use any-other antipiracy strategies.
Information about activated serials numbers and installations will be stored in MySQL database accessible with standard DB tools.
Collection of serial numbers will be generated by concatenation of product version and a random gamma. So each key will have the following structure:

SEAG-VVVV-RRRR-RRRR-CCCC,

where VVVV- product version number, RRRR – random gamma, CCCC – serial control sum for typo prevention.

1.4.12.2 Activation schema

To enable paid functionality user must pass through the product activation procedure. The application will provide an activation dialog to enter product serial number. Then the number with information unique identifying user's computer (MAC address is a good example) will be passed to the license server.
The server will check validity of the license request. In a positive case it will encrypt the license request with a private key by RSA algorithm and send it back to the application.
The application will store the signed license. On each startup the app will try to decrypt the license and check validity of the environment. If the environment will match the license, paid functionality will be enabled.

1.4.13 Installation

The application will be installed in an easy way with native platform installer.

1.4.13.1 Required permissions

The application must not require administrative (superuser) rights to install correctly.

1.4.13.2 Runtime environment

1.4.13.2.1 Java

Java 1.4.2 shall be used to implement the application. This is the lowest common denominator for available platforms.

1.4.13.2.2 Operating systems

1.4.13.2.2.1 Windows

The application must run on the following Windows versions: 2000, XP, Vista, 2003.

1.4.13.2.2.1.1 Installer

NSIS based self-extractable installer will be used.

1.4.13.2.2.1.2 Uninstall

The installer will register Ignoble in the Windows application registry to allow uninstall via Windows->Control panel->Add/Remove programs applet.

1.4.13.2.2.1.3 Java Runtime Environment

The installer will be bundled with JRE environment needed to run the application. The bundled JRE will be installed.

1.4.13.2.2.2 MacOS X

The application must run on MacOS X 10.4 and better.

1.4.13.2.2.2.1 Installation

a native dmg image will be used to distribute application. The application will be installed by a simple drag'n'drop operation.

1.4.13.2.2.2.2 Uninstall

User will be able to install application just be moving application package (icon) into the trash.

1.4.13.2.2.2.3 JRE Environment

Installed OS X JVM will be used to run the application.

1.4.14 Out of the project scope

1.4.14.1 Dreamweaver extension

Due the nature of the project technology profile the extension will be deferred until the first release of the application.

1.5 Application architecture

The application will be built on top of Eclipse Rich Client Platform infrastructure. The platform provide rich UI toolkit for development of cross-platform extensible desktop applications. So it is a natural to use it as a foundation for the Ignoble application.

1.5.1 Data structure

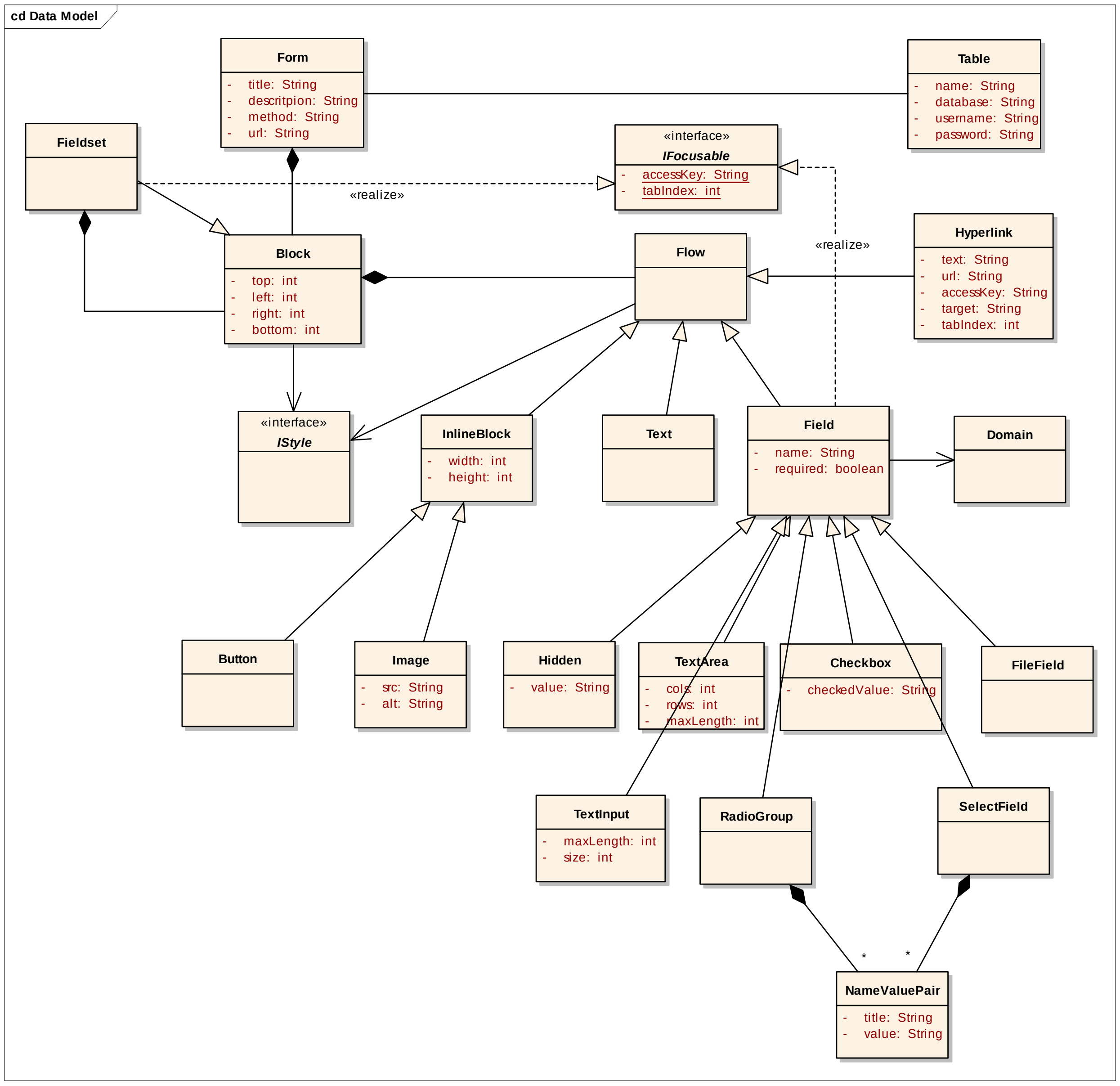

Fig. 1 Data model structure

1.5.2 Component break-down

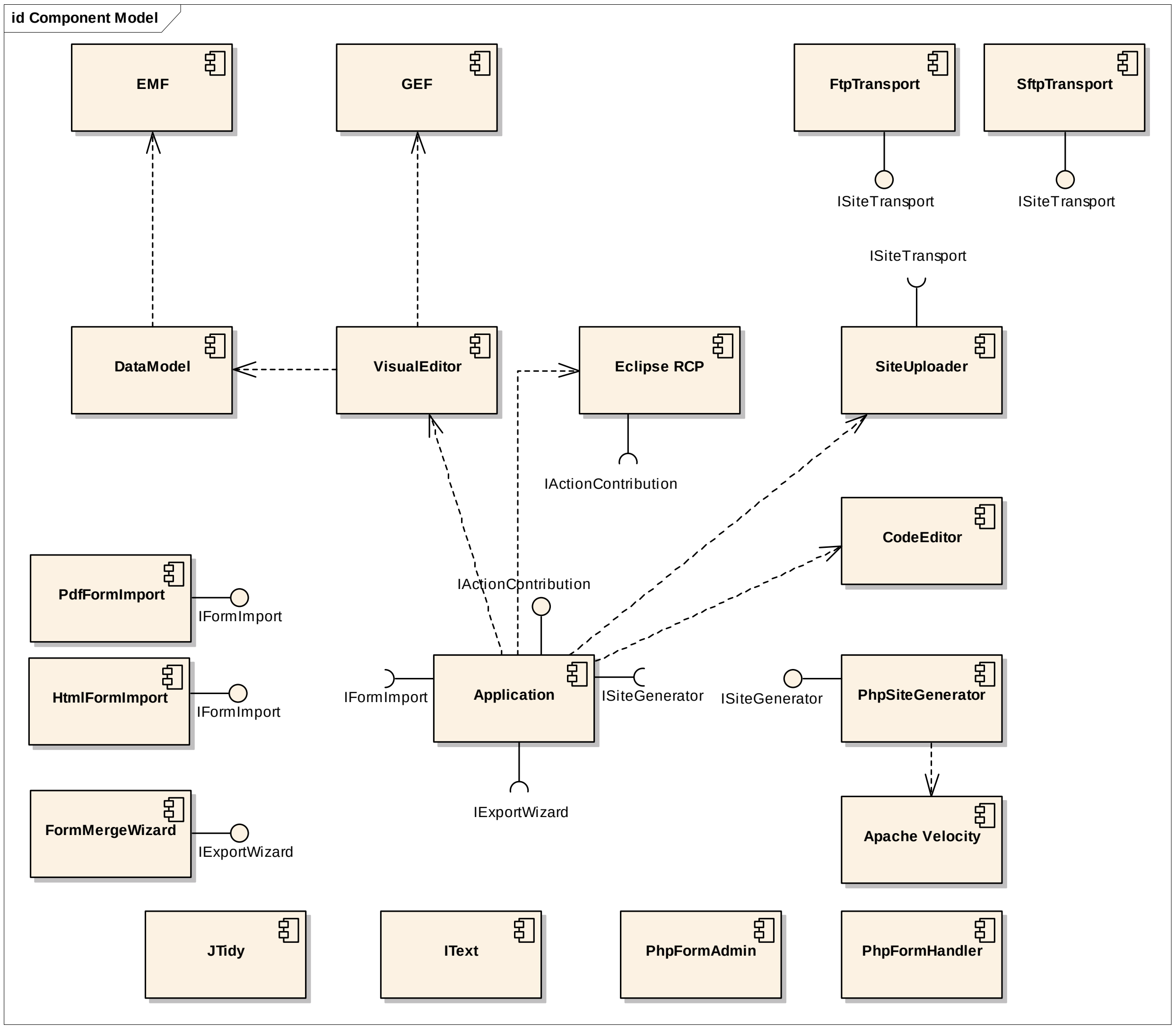

Fig. 2 Component break-down

On the Fig. 2 you can see component structure of the application. Each component of the diagram represents Eclipse bundle (feature, plugin or fragment). The following system extension points introduced:

Interface	Purpose
IActionContribution	Addition of new actions into application's main menu bar or main toolbar.
ISiteGenerator	Addition of new form target environments.
IFormImport	Addition of new imported form formats
IExportWizard	Addition of new form export wizards
ISiteTransport	Addition of new form upload protocols

1.5.3 Third-party components

Title	Description	Version	License
XStream	XML serialization/deserialization		BSD
Eclipse RCP	Application framework	3.3M4	EPL
Eclipse GEF	Graphical editing framework	3.2.1	EPL
Amateras HTML editor	Advanced HTML editor		CPL
Jazzy	Spell checker		LGPL
Commons-Net	FTP transfer		Apache
JCraft Jsch	SFTP transfer		BSD
iText	PDF reading		MPL
NSIS	Windows Installation builder		GPL
FileStorm	MacOS X DMG image builder		Commercial

1.6 References

[DDS] Detailed description of Ignoble, Great Software Inc, December 2006

Communications Plan:

Collaboration space:

This project will utilize a collaboration space at url: http://crm.swxperts.com/clientportal
This space will be accessible through internet.
The space will be used to store:

- Project Status summaries
- Action item lists
- Any project review documents that the core team needs access to
- Other information

Communications

Project communication includes phone calls, faxes, emails, Instant Messaging etc. The primary communication collaboration path will be Project Manager -Customer. The media and tools required for the type of communication are listed below:

- **Analyst to Customer**

Media/Tools: Project Manager and Customer must maintain access to the following media and tools: email, IM (Yahoo) and Microsoft Word.
Ground rules: Project Manager and Customer will check e-mail daily and reply within 12 hours. Project Manager and Customer are available for online communications on mutually agreed hours.
Project Manager is accessible via Yahoo at
Customer is accessible via Yahoo at

Email distribution list

Customer and Project team will communicate via
Project Manager Alexey **Bubba– ab@swxpwerts.com**
Account Manager Brent Coats– ab@swxpwerts.com
Customer Carl Dresden - **ab@swxpwerts.com**

Weekly status reports and billing

Responsibility and Format: Account Manager will generate status reports and invoices for work performed and email it to Customer and all participants from the distribution list.
Who receives the communication: Customer will have access to these reports and receive invoices.
Media/Tools: Deliverables are published in Microsoft Word or other media or tool specified by Customer. Invoices are published in Adobe PDF.
Ground Rules: Invoices are to be paid upon receipt.

Deliverables & Key Milestones

Project Start Date: 02/19/2007Project End Date: 10/17/2007
Project Budget: $53736

Task title	Deliverables	Date	Comments
Milestone 1	Application core	03/30/2007	A basic Eclipse application running on both Windows and OS X platforms will be built. The application will allow editing a form by placing/moving/deleting the form blocks. Also Open/Save/Save As functionality will be implemented. During the milestone an automatic build/test system will be setup. This build server will run on a daily basis as the minimum to enable for daily tracking of the project progress.
Milestone 2	Form Editor	05/29/2007	Will include almost completed form editor with support of inline and block layouts together with HTML input controls. Also Amateras HTML code editor will be integrated into the system. However, it will not be possible to generate form code.
Milestone 3	Integration Release	07/25/2007	The application will enable generating functioning form from the graphical representation and edit it in the code editor. The application framework will be enhanced to have the visual design with global wizard as described in the original SRS specification. Also all HTML controls will be available in the graphical form editor.
Alpha Testing	Alpha Release	09/07/2007	The code stabilization release. As the result of the milestone the release ready for external testing shall be build. Intensive testing/bug fixing in these release will be done. Also Alpha is the last release before the public release that will be able to accommodate new functionality.
Beta Testing	Beta Release	10/02/2007	This phase implies in-life product testing by loyal third-party testers. This pursues two things: the primary goal is to test the product in the real-life in order to identify some problems with compatibility/installation & etc. The second goal is promotion. As the result of this development phase a product quality release will be built.
Final Release	Final Release	10/17/2007	Final release of the ready product as per the specifications.

Payment Schedule

Upfront payment	02/19/2007 - $4500
Milestone 1	03/30/2007 - $8956
Milestone 2	05/29/2007 - $8956
Milestone 3	07/25/2007 - $13412
Alpha	09/07/2007 - $8956
Final payment upon Release	10/17/2007 - $8956

Invoices are to be paid upon receipt.

Great Software, Inc.	**Software Experts, LLC**
Name:________________________.	**Name: Paul Swengler .**
By: ___________________________.	**By:__________________________.**
Title: Owner .	**Title: CEO .**