This document describes the process, milestones and required deliverables for your team project in CS 499, Fall 2008. You may begin working on the project as soon as your team is formed.

1 Deliverables

The following are the major required deliverables. Every team must turn in each of these documents (one for the whole team) at some point. It is strongly recommended that you submit draft versions for feedback along the way.

Project Selection. (5 points.) Specifies what project you will create, and who is the customer. Due 23 September.

Project Plan. (15 points.) Describes the development process you have chosen, along with the tools, languages, and pre-existing software you intend to use. It is highly recommended that you also lay out roles and responsibilities for each team member. Finally, this document states exactly what your team will deliver (to me), and when. Due 2 October.

Requirements Analysis/Specification. (20 points overall.) Unambiguously describes the required behavior and features of the system to be implemented. Must be approved by your customer at some point in your process; a final version must be submitted before your project presentation/demonstration. The due date for this will depend on the process you intend to use. For some approaches, you will go through multiple iterations on this.

Design Document. (15 points overall.) Describes the modules of the system, the interfaces between them, and their functional responsibilities. Describes the major data structures to be used, as well as the information flow in the system. The due date for this will depend on the process you intend to use. For some approaches, you will go through multiple iterations on this.

Test Plan. (20 points overall.) Describes unit and integration testing scenarios/cases, and how they will be tested.

Project Presentation/Demonstration. (25 points.) Every team will give a 30-minute presentation of its project, including a live demonstration of its capabilities, if possible. Each team member is expected to participate in the presentation. Due 9 or 11 December.

Weekly Team Progress Reports. Each team submits one of these each week except the week of Thanksgiving. They may be emailed, but must be “postmarked” (i.e., it must be in my inbox) before midnight each Thursday unless you make other arrangements with me. Failure to submit a weekly team progress report will result in lowering of your team’s overall grade by 3% for each report not submitted. There will be 11 weeks between team formation and
presentations (Thanksgiving week is excluded.) So if you submit no progress reports, your team’s maximum possible score on the project will be 67.

Remember that **neatness and appearance counts** when it comes to deliverables. Also, **all deliverables must be provided in hard copy** except for weekly progress reports, which may be emailed in electronic form.

2 Timeline

Apart from the above-specified deadlines, there will be three **Project Milestones**, one every three weeks beginning Tuesday, 14 October. Your project plan will specify what you will deliver at each milestone. If you are using a linear, or “waterfall” process, you will probably deliver the final version of each of the deliverables at one of the milestones. If you are using an iterative process, you may deliver a draft version of all of the deliverables on the first Milestone date, and refined versions at each subsequent date.

Here’s a handy clip-n-save timeline for future reference:

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue 9 Sep</td>
<td>Teams formed</td>
</tr>
<tr>
<td>Thu 18 Sep</td>
<td>Weekly progress report due each Thu thru 4 Dec</td>
</tr>
<tr>
<td>Tue 23 Sep</td>
<td>Project selection due</td>
</tr>
<tr>
<td>Thu 2 Oct</td>
<td>Project plan due</td>
</tr>
<tr>
<td>Tue 14 Oct</td>
<td>Milestone I</td>
</tr>
<tr>
<td>Tue 4 Nov</td>
<td>Milestone II</td>
</tr>
<tr>
<td>Tue 25 Nov</td>
<td>Milestone III</td>
</tr>
<tr>
<td>Tue 9 Dec</td>
<td>Presentations 1 &amp; 2</td>
</tr>
<tr>
<td>Thu 11 Dec</td>
<td>Presentations 3 &amp; 4</td>
</tr>
<tr>
<td>15-19 Dec</td>
<td>Exit interviews w/Calvert</td>
</tr>
</tbody>
</table>

3 Project Parameters

The following are absolute requirements for an acceptable project:

1. **Nontrivial.** The project must require a non-negligible level of effort for a four-person team over a 10-week period.

2. **Feasible.** There must be a reasonable chance that the project can be successfully carried out, given a due level of diligence, according to the above timeline.

3. **Novelty.** The project involve building something no team member has built before.

4. **Stakeholder.** A suitable person outside the team must be identified to act as the “customer” for the system.

5. **Software.** The project must require the creation of some amount of new software for its implementation. “Software” here is loosely defined, but simply configuring existing software does not count.
4 Project Possibilities

What follows are the current projects I have. Alternative suggestions are still possible, but not for much longer. Note that inclusion in this list does not necessarily imply that the project satisfies all the requirements above. If your team is interested, I can provide additional information or put you in touch with the stakeholder.

4.1 Online Scheduling Request Program

**Stakeholder:** Brian Dixon, MD  
**Context:** University of Kentucky Medical School  
**High-level Description:** On-line system for residents and attending physicians to request changes to their daily on-call schedule, have those changes approved, and update the master schedule.

**Details:** Currently the chief resident makes a schedule in paper form each year; the schedule shows who is on call each day (residents and attendings) at each clinical site throughout Lexington. Whenever a resident wants to request a change (vacation, illness, etc.) it requires the requestor to print off a “schedule change request form”, fill it out, and take it to every person involved for approval. This includes: the attending physician at the site they will be absent from; the resident who will cover the absent resident; the program director; the program administrative assistant who maintains the master schedule; the Chief Resident who must ensure that work requirements are equal; and the scheduling person at the site, who may need to reschedule patients’ appointments. This alone makes it a significant problem, but when you add in the problem different change requests (possibly involving overlapping sets of people) being processed in parallel, it is really “a nightmare”.

What is desired is an online, web-based solution that would allow a resident requesting a change to fill out a form, indicating who needs to be notified. The system would then email a notification to the involved parties, who would then log onto the internal web site and approve (or not) the request. Once all approvals are obtained, the system would update the schedule and notify all the necessary parties that the change is official.

The system should take steps to ensure that concurrently-requested changes do not stomp on each other, and that the schedule remains consistent at all times.

4.2 Adword Auction Bidding Agent Framework

**Stakeholder:** Prof. De Liu, Business and Economics  
**Context:** Auctions for Google Adwords  
**High-level Description:** Build a framework for testing and evaluate algorithms for “intelligent bidding agents” in “pay-per-click” adword auctions.

**Details:** Google and other search engines auction off the rights to have advertisers’ links appear above the top results when users search for particular keywords or combinations (e.g., “toyota”, “soccer shinguard”, “gas dryer”, etc.). It is an active area of research to develop optimal bidding strategies for use in such auctions. The idea of this project is to develop a simulation of an online adword auction framework for evaluating bidding strategies. (Microsoft held an online contest among such bidding agents in 2006.) The framework would produce (simulated) data about which
searches result in clicks on advertisers’ links, and how much money was paid and made. The agent
must have a well-defined interface to a bidding policy, which could be changed, either by plugging
in a completely new object, or by adjusting parameters.

4.3 Ride-Matching System

Stakeholder: TBD (Prof. Hayes?)
Context: Commuters from rural areas into metropolitan areas.
High-level Description: Build a system (presumably web-based) for matching up people who
commute into a metropolitan area from outside. The system must consider security and privacy.
Ideally, input as well as output could be done using, e.g., Google maps data.

4.4 Medical Claim EDI to PDF

Stakeholder: Dr. Tony Baxter/CorrectCare
Context: Medical Billing Systems
High-level Description: Build a system to convert medical forms encoded using ANSI X12 (an
EDI—electronic data interchange—protocol) into an industry-standard paper format for humans
to view and/or print.
Details: Most medical claims are generated and submitted electronically using the X12 EDI
format. This is great for reducing transcription errors but is very difficult to troubleshoot when
things go wrong. There exist industry standard paper forms, which all claims processors are
familiar. The ability to translate X12 claims into PDF for these standard forms (called UB-04
and HCFA-1500) would be a powerful first-stage troubleshooting tool. It could also be useful for
presenting claim information to patients. This project would involve learning about the ANSI X12
EDI standard and producing a translation system. The ANSI X12 formats are described using a
finite state machine, which could be the basis for a lexical analyzer and parser.

4.5 Medicaid Repricer

Stakeholder: Dr. Tony Baxter/CorrectCare
Context: Medical Billing Systems

High-level Description: Build a system for adjusting medical claim costs for standard services
to reg
Details: It is common for states to tie their medical reimbursements for medical claims for inmates
to Medicaid accepted values. Usually, this is something like a percentage (e.g., 120% or 130%)
of regionally accepted rates. A physician or hospital (the provider) will submit a claim and it is
repriced to the appropriate rate. As a contractor to the Ohio Department of Rehabilitation and
Correction (ODRC), CorrectCare performs this repricing service. The current 3M software used
to reprice these medical claims is geared towards hospital use and works well if used for a specific
hospital. It does not integrate well into CorrectCare’s workflow, where they provide services for
a large number of different hospitals. They need software that easily switches between hospitals
and retains the values from previously repriced claims. Merely revising the existing software is
not an option as it was commercially purchased. The code is proprietary, and the source code is unavailable.

The project would involve some database design and implementation, a small amount of reporting, and analysis and implementation of Medicaid pricing.

### 4.6 Web-based Map Editor

**Stakeholder:** TBD  
**Context:** Web-based services

**High-level Description:** Build a web-based map editing system that allows people to create, edit, and collaborate on web-based maps.

### 4.7 Course Schedule Validator

**Stakeholder:** Dr. Jerzy Jaromczyk, UK CS  
**Context:** UK administration

**High-level Description:** Build a system to take output from the UK course/schedule database and detect conflicts between courses.

### 4.8 Bike Key Checkout System

**Stakeholder:** Prof. Calvert  
**Context:** Physical Resource Security

**High-level Description:** Build a system to allow a community of users to check out a resource that is secured using a traditional lock and key. The idea is that the system always knows who checked out the key last.

**Details:** The idea is a generalization of to the “lockboxes” used by real estate agents: the house key is locked in a box, which is secured by a combination that (in theory) only real estate agents know. The problem with that system is that you have to trust the agents. The desired system here would have a lockbox that could be opened (only) under computer control. A web interface would authenticate the user desiring to check out the key, using strong cryptographic authentication (or perhaps fancy biometrics), record the transaction, then open the box. The system would need to be able to verify that the key was actually physically present (e.g., by inserting it in a lock equipped with sensors) before crediting the borrower with returning it.

This project would involve at least the specification of specialized hardware, and the development of software to control it, along with a web interface to that software. Possible enhancements would include a scheduling/reservation subsystem.