Scenario #1 Online Registration System

Problem Statement

As the head of information systems for a college you are tasked with developing a new student registration system. The college would like a new client-server system to replace its much older system developed around mainframe technology. The new system will allow students to register for courses and view report cards from personal computers attached to the campus LAN. Professors will be able to access the system to sign up to teach courses as well as record grades.

Due to a decrease in federal funding, the college cannot afford to replace the entire system at once. The college will keep the existing course catalog database where all course information is maintained. This database is an Ingres relational database running on a DEC VAX. Fortunately the college has invested in an open SQL interface that allows access to this database from college's Unix servers. The legacy system performance is rather poor, so the new system must ensure that access to the data on the legacy system occurs in a timely manner. The new system will access course information from the legacy database but will not update it. The registrar's office will continue to maintain course information through another system.

At the beginning of each semester, students may request a course catalogue containing a list of course offerings for the semester. Information about each course, such as professor, department, and prerequisites, will be included to help students make informed decisions.

The new system will allow students to select four course offerings for the coming semester. In addition, each student will indicate two alternative choices in case the student cannot be assigned to a primary selection. Course offerings will have a maximum of ten students and a minimum of three students. A course offering with fewer than three students will be canceled. For each semester, there is a period of time that students can change their schedule. Students must be able to access the system during this time to add or drop courses. Once the registration process is completed for a student, the registration system sends information to the billing system so the student can be billed for the semester. If a course fills up during the actual registration process, the student must be notified of the change before submitting the schedule for processing.

At the end of the semester, the student will be able to access the system to view an electronic report card. Since student grades are sensitive information, the system must employ extra security measures to prevent unauthorized access.

Professors must be able to access the on-line system to indicate which courses they will be teaching. They will also need to see which students signed up for their course offerings. In addition, the professors will be able to record the grades for the students in each class.

Possible Technical Risks

Risk#	Description
1	SQL injection allows the hack to login the system by inputting specifically crafted SQL commands with the intent of bypassing the login form barrier and seeing what lies behind it.
2	Networking failure.
3	Web Server could not handle amounts of requests at the same time.
4	Database could not process many transactions at the same time.
5	Students could access to the grading system.
6	The student could not be notified of the change before submitting the schedule for processing when a course fills up during the actual registration process,
7	The requests and response are very slow.
8	When the registration process is completed for a student, the registration system could not send information to the billing system.

Risk Mitigation/avoidance strategies

Risk#	Mitigation / Avoidance Strategies
1	Don't use dynamic SQL when it can be avoided; Use appropriate privileges; Apply patches and updates as soon as practical; Don't divulge more information than you need to.
2	Provide a good fault-tolerant system and back-up in time.
3	Deploy distributed system.
4	Devise the database more wisely and build reasonable mechanism
5	Secure the privilege mechanism.
6	Deal with the concurrency correctly.
7	Use load balancer correctly.
8	Apply design pattern and create objects correctly.

Iteration Planning

Use case included

1. Student registration

Primary Actor: Student

Pre-conditions: Student must be admitted to college and sign up for the system and assigned a user ID and password.

Post-conditions:

- Student will be registered for at least four courses offered by the college.
- Student will be allowed to register two more courses in case the student cannot be assigned to a primary selection.

Basic Flow or Main Scenario:

- Sign up an account for registration system with student ID and password
- Activate the student account

Extensions or Alternate Flows: None

Exceptions:

- Non-activated user id
- Pre-requisite courses not met
- No permission to register a class
- Registration fails if bill is not paid before due
- Course catalog unavailable

Related Use Cases: View report cards

2. Professor operation

Primary Actor: Professor

Pre-conditions: Professor must have taught at least one course at the college

Post-conditions: Professor must report grades for students.

Basic Flow or Main Scenario:

- Sign up an account for registration system with Professor ID and password
- Activate the Professor account

Extensions or Alternate Flows: None

Exceptions:

1. Non-activate Professor account

2. Course taught not in course catalog

Related Use Cases: Professor grading

3. Registration System Management Use Case

Primary Actor: Registrar

Pre-conditions: Registrar needs Registrar privileges to manage this Registration System through Registrar's ID and password.

Post-conditions: Professors, students, courses and grade records will be updated.

Basic Flow or Main Scenario:

- Enter Professor ID and password
- Validate the Professor ID and password
- Select the upcoming academic semester-year
- Display all courses to teach
- Select a course
- Display names of students who signed up for this course
- Display the number of students who signed up for this course

Extensions or Alternate Flows: None

Exceptions:

1. Student enrolls in the class which exceeds the maximal capacity or does not the minimal capacity.

2. Student drops class after the registration class due.

3. Professor teaches the class which exceeds the maximal capacity or does not the minimal capacity.

4. Professor could teach class after retirement or separation.

Related Use Cases: View Course Status to teach

Iteration Plan

Iteration 1: It takes 20 days

Task	Task Description	Start	End	Resources
General	Write/Update Software Requirement Specification			RequisitePro, Microsoft office, JUCMNAV, Sparx System's Enterprise Architect
Client/Server Prototyping	Draw Blueprints			
	Write UML Use Cases and Draw Use Cases Diagrams			
	Draw Sequence Diagrams			
	Design the Prototype			
Management	Write a Iteration Plan			
	Allocate Tasks			
Review	Review Previous Work and Make a Conclusion			

Iteration 2: It takes 30 days

Task	Task Description	Start	End	Resources
General	Draw Data Flow			RequisitePro, Microsoft office, JUCMNAV, Sparx System's Enterprise Architect, Cisco Networking, Server, PCS.
	Update UML Use Cases and Use Cases Diagrams			
	Update Sequence Diagrams			
	Update/Revise Prototype			
	Build Architecture			
	Devise Design Pattern			
	Design Class Diagrams and Relation			
Implementation	Deploy Networking, Server and Database			
	Code for Basic Classes			
Management	Check Progress and Clarify Tasks			
Review	Review Previous Work and Make a Conclusion			

Iteration 2: It takes 60 days

Task	Task Description	Start	End	Resources
General	Check Codes and Fix Bugs			RequisitePro, Microsoft office, JUCMNAV, Sparx System's Enterprise Architect, Eclipse, jUnit.
	Devise Test Cases			
Implementation	Code for All Modules			
	Finish All Functions			
	Code for Tests			
Management	Check Progress			
	Verify Test Results			
Review	Review Previous Work and Make a Conclusion			

Software architecture, sequence diagrams and class diagrams should take place in earlier iterations, because it provides a blueprint and direction to follow.

Coding and tests should take place later iterations. Coding will ensure the project could be finished in time and tests could make sure the correctness of the code.

Iterations 2 and Iterations 3 should address the most architecturally significant use cases.

In order to reduce risks the iteration of requirement analysis and the system architecture design should be done earlier so that the system architecture could be identified earlier which is very important for the later iteration of each part of the architecture.

Scenario #2 Mine Pump Control System

Problem Statement

You have been tasked to develop the architecture for a mine pump control system, designed to monitor and pump flood water out of mine shafts. As underground mining operations take place far below the water table, flooding into mine galleries and shafts is an ever-present danger. Excessive flooding is clearly a safety hazard for workers, but also has profitability implications ranging from equipment damage to productivity delays, to mine closures in extreme circumstances.

The system to be developed will be required to monitor the water level in a given mine shaft using two sensors. A high water sensor that measures the maximum acceptable level of flooding in a shaft before pumping begins, and a low water sensor, which measures the minimum level of acceptable flooding and pumping stops. These sensors are used to start a mine pump. When the flooding level exceeds the level determined by the high water sensor the pump is switches on. When the water has been pumped out and the minimum level of acceptable flooding has been reached, as measured by the low water sensor, the pump switches off.

In addition to flooding mining is often hindered by methane pockets, where gas seeps into the shafts and galleries triggering an evacuation. Again this is a safety hazard, the mining staff won't be able to breathe, and even more critically, operating equipment may generate sparks which will cause the methane to ignite.

Therefore the system will include a methane sensor that will be used to trigger an evacuation alarm in the presence of dangerous levels of methane (measured in N parts per million), and also switch off the pump regardless of the current water level.

The system is used by two key roles. The first is the Operator. This role is required to log in to the system with a username and password. Following a successful login the operator is able to start or stop the pump if, and only if, the water level is between the high and low sensor limits. The second role is the Supervisor. A supervisor must verify their security credential as per the operator above. Following a successful login they are able to switch the pump on, or off at any time. For example a supervisor could run the pump after the flood level has dropped below the level set by the low water sensor. They could also switch the pump off if the water level goes over the maximum high level of flooding. In these cases the supervisors' actions override the automatic behavior of the pump. A supervisor is required to "reset" the pump system in order to re-establish automatic behavior.

Finally to meet Federal monitoring standards a persistent log is required to capture the following events:

Pump switched on by high water sensor
Pump switched off by low water sensor
Pump switched on or off by operator or supervisor
Evacuation alarm triggered by methane sensor
The reading of the methane sensor every 30 minutes

The reading of the methane sensor (for the last 24 hours) can be read by the operator. All readings (up to 30 days) can be read by the supervisor. The supervisor also has the capability to add a "note" to any specific log event that occurs within 24 hours.

Possible Technical Risks

Risk#	Description
1	SQL injection allows the hack to login the system by inputting specifically crafted SQL commands with the intent of bypassing the login form barrier and seeing what lies behind it.
2	Networking failure.
3	Web Server could not handle amounts of requests at the same time.
4	Database could not process many transactions at the same time.
5	Operators could switch the pump on, or off at any time.
6	The log could not capture required events because other components suddenly crush.
7	The requests and response are very slow.
8	There may be so many duplicate Objects when there are many requests and it may cause a system crash

Risk Mitigation/avoidance strategies

Risk#	Mitigation / Avoidance Strategies
1	Don't use dynamic SQL when it can be avoided; Use appropriate privileges; Apply patches and updates as soon as practical; Don't divulge more information than you need to.
2	Provide a good fault-tolerant system and back-up in time.
3	Deploy distributed system.
4	Devise the database more wisely and build reasonable mechanism
5	Secure the privilege mechanism.
6	Deal with failure events correctly.
7	Use load balancer correctly.
8	Apply Singleton Design Pattern, Factory Pattern if it is possible which produces reusable Objects and avoid creating duplicate and useless instances.

Iteration Planning

Use case included

1. Operator Management in Water Sensor

Primary Actor: Operator

Pre-conditions: Operators must be permitted to operate the system and assigned a username and password.

Post-conditions: Operator could observe the status of the high and low sensor, and start/stop the pump when water level is between the high and low sensor limits

Basic Flow or Main Scenario:

- Enter Operator’s username and password
- Validate the username and password
- Observe the status of the high and low sensor
- Start/Stop the pump only when water level is between the high and low sensor limits

Extensions or Alternate Flows: None

Exceptions:

- Non-activated username or password
- Start/Stop the pump when water level is not between the high and low sensor limits
- Start/Stop function is not available

Related Use Cases: The Reading of Methane Sensor by Operator

2. Supervisor Management in Water Sensor

Primary Actor: Supervisor

Pre-conditions: Supervisor must be permitted to operate the system and assigned a username and password.

Post-conditions: Supervisor could observe the status of the high and low sensor, and start/stop the pump at any time

Basic Flow or Main Scenario:

- Enter Supervisor’s username and password
- Validate the username and password
- Observe the status of the high and low sensor
- Start/Stop the pump any time

- Reset the pump system in order to reestablish automatic behavior

Extensions or Alternate Flows: None

Exceptions:

- Non-activated username or password
- Start/Stop function is not available
- Reset function is not available
- Supervisors' actions could not override the automatic behavior of the pump

Related Use Cases: The Reading of Methane Sensor by Supervisor

3. Reading of Methane Sensor by Operator

Primary Actor: Operator

Pre-conditions: Operators must be permitted to operate the system and assigned a username and password.

Post-conditions: Operator could observe the level of methane for the last 24 hours

Basic Flow or Main Scenario:

- Monitor the level of methane (measured in N parts per million)
- Trigger an evacuation alarm in the presence of dangerous levels of methane
- Switch off the pump regardless of the current water level

Extensions or Alternate Flows: None

Exceptions:

- Non-activated username or password
- Methane sensor is not available
- Read the methane sensor prior to 24 hours
- Trigger an evacuation alarm not in the presence of dangerous levels of methane

Related Use Cases: Methane Sensor Alarm

4. Reading of Methane Sensor by Supervisor

Primary Actor: Supervisor

Pre-conditions: Supervisor must be permitted to operate the system and assigned a username and password.

Post-conditions: Supervisor could select a period to read methane sensor up to 30 days

Basic Flow or Main Scenario:

- Enter Operator's username and password
- Validate the username and password
- Select the period to read (for the last 24 hours)
- Read the methane sensor

Extensions or Alternate Flows: None

Exceptions:

- Non-activated username or password
- Methane sensor is not available
- Read the methane sensor prior to 30 days
- Trigger an evacuation alarm not in the presence of dangerous levels of methane

Related Use Cases: Methane Sensor Alarm

5. Log Management System

Primary Actor: Supervisor

Pre-conditions: Supervisor must be permitted to operate the system and assigned a username and password.

Post-conditions: Supervisor could add a note to a specific log

Basic Flow or Main Scenario:

- Enter Supervisor's username and password
- Validate the username and password
- Select period to read (up to 30 days)

- Read the methane sensor
- Select a log event that occurs within 24 hours to add a note
- Add a note to this log event

Extensions or Alternate Flows: None

Exceptions:

- Non-activated username or password
- Adding a note by supervisor is not available
- Capturing events is not available

Related Use Cases: Reading of Methane Sensor

Iteration Plan

Iteration 1: It takes 20 days

Task	Task Description	Start	End	Resources
General	Write/Update Software Requirement Specification			RequisitePro, Microsoft office, JUCMNAV, Sparx System's Enterprise Architect
Client/Server Prototyping	Draw Blueprints			
	Write UML Use Cases and Draw Use Cases Diagrams			
	Draw Sequence Diagrams			
	Design the Prototype			
Management	Write a Iteration Plan			
	Allocate Tasks			
Review	Review Previous Work and Make a Conclusion			

Iteration 2: It takes 30 days

Task	Task Description	Start	End	Resources
General	Draw Data Flow			RequisitePro, Microsoft office, JUCMNAV, Sparx System's
	Update UML Use Cases and Use Cases Diagrams			
	Update Sequence Diagrams			
	Update/Revise Prototype			

	Build Architecture			Enterprise Architect, Cisco Networking, Server, PCS.
	Devise Design Pattern			
	Design Class Diagrams and Relation			
Implementation	Deploy Networking, Server and Database			
	Code for Basic Classes			
Management	Check Progress and Clarify Tasks			
Review	Review Previous Work and Make a Conclusion			

Iteration 2: It takes 60 days

Task	Task Description	Start	End	Resources
General	Check Codes and Fix Bugs			RequisitePro, Microsoft office, JUCMNAV, Sparx System's Enterprise Architect, Eclipse, jUnit.
	Devise Test Cases			
Implementation	Code for All Modules			
	Finish All Functions			
	Code for Tests			
Management	Check Progress			
	Verify Test Results			
Review	Review Previous Work and Make a Conclusion			

Software architecture, sequence diagrams and class diagrams should take place in earlier iterations, because it provides a blueprint and direction to follow.

Coding and tests should take place later iterations. Coding will ensure the project could be finished in time and tests could make sure the correctness of the code.

Iterations 2 and Iterations 3 should address the most architecturally significant use cases.

In order to reduce risks the iteration of requirement analysis and the system architecture design should be done earlier so that the system architecture could be identified earlier which is very important for the later iteration of each part of the architecture.