TIM 58 Project: Group 6

By
Allison Rogan
Ethan Cox
Ekra Haq
Jacob Farinha
Chudi Atuegbu
Justin Cheng

Table of Contents

Executive summary:

For efficiency and cohesiveness, a Service Personnel Management System will create a wholesome process in which service engineers, administrative staff, and clients are connected. Whether external or internal, all users of the system can have access to information when an action takes place from an engineer or client.

By creating an efficient system that oversees its users, there will be increased interaction and coordination, while minimizing conflicts. Transparency of information as well as ease of access is key, as there are numerous engineers, technicians, supervisors, and programmers that each have differing but critical aspects to their roles. System processes can be seen by administrative and higher level employees, while service engineers are able to relay progresses of work orders to create a system that does not require constant supervision.

Additionally, with reduced supervision, more resources can be devoted towards training, customer satisfaction, and hiring. An essential aspect of the Service Personnel Management System is a 24/7 uptime, so that at any moment, responses and work orders are available on the server. The development team will be extremely concise in ensuring that all external and internal functions are quick, simple, but encompassing of the many functions and processes of its users.

Systems request

Business Requirements: Personnel Management System connecting the functions of external and internal users to improve response times and minimize costs, while having a consistent and familiar interface.

Business Need: To assist service engineers working on customer repairs or installations through a management system that coordinates scheduling and training.

Business Value:

We expect our Service Personnel Management System to improve customer experience through a more time-sensitive and cost-effective management system that assists with maintenance processes between customers and service engineers. Over the long run, this will prove to be more profitable for companies who use our system because of improved efficiency.

Special issues or constraints:

1. The system should be designed and implemented within 1 year in order to provide enough time for bug fixes or other improvements
2. We will need to hire a team of highly-skilled engineers to develop our SPMS
3. The system will need to be tested before officially released

Feasibility Analysis

Guides the organization to determine whether or not to proceed with the project. This also identifies the significant risks associated with the project that must be brought forward if the project does get approved.

- Technical Feasibility: Can we build it?
- Familiarity with functional area, technology: Though our IT department is highly skilled and familiar with database systems from work in the past, there is still some risk with this project given that it is the first of its kind that we have created.
- Project size: Our project is considerably large and will require a great deal of time, research and tests so there is a very high risk.
- Compatibility: There is somewhat high risk because we are creating a new type of system, but our IT personnel are experienced in database and app development.
- Economic Feasibility: Should we build it?
- Development costs: focusing on the development team salaries, consultant fees, hardware/software etc.
- Annual operating costs: includes software upgrades, licensing fees, hardware repairs/upgrades
- Annual benefits (cost savings and revenues): includes increased sales, reductions in staff/inventory/IT costs
- Intangible costs and benefits: Training costs, software development costs
- Organizational Feasibility: If we build it, will they come?
- Is the project aligned with the business? Yes because we are developing an IT system
- Project champion(s): Project managers
- Senior management: CEO, CFO, board of directors
- Users: Customers who are in need of service repairs
- Other stakeholders: stockholders

Requirements Definition

❖ Requirements determination is required to transform the system request's business requirements into a more specific list of what the new system must do to. Overall a requirement is just a statement of what the system must do or what else it needs to have.

❖ **During a systems development project:**

➢ Business requirements:

        - These requirements are created to tell or describe to us of what the business needs.
    - ➢ User requirements:
        - Tells what the users needs to do.
    - ➢ Functional requirements:
        - Provides what the software should do.
    - ➢ Non Functional Requirements:
        - Provides what the characteristics the system should have.
    - ➢ System requirements:
        - Shows how the system should be built.
- ❖ **How to gather a list of requirements:**
    - ➢ Interviews (five step process)
        1. Selecting interviewees:
            a. A list of who will be interviewed
        2. Designing interview questions:
            a. If you need to know something, you ask someone.
        3. Preparing for the interview:
            a. Interviews are one on one, but sometimes several people are interviewed at the same time.
            b. Schedule; where and when it will take place
        4. Control and manage the interview:
            a. Our Project sponsors, business users, and project team members can help the analyst determine who in the organization can best provide important information about requirements.
        5. Post and share that following interview:
            a. People are selected based on what information the analyst's needs.
- ❖ **User stories for the system.**
    - ➢ As a user I want to create a service order so that my equipment will be fixed.
    - ➢ As a programmer I want to create the system so it searches for the nearest qualified service engineer, and schedules them to fulfill the service order.
    - ➢ As a service engineer I want to record the completion of a service order so the company knows it has been fulfilled.
    - ➢ As a scheduler I want to create new courses and instances of those courses so that service engineers can upgrade their skills.
    - ➢ As a programmer I want to create the system so that it updates registration in courses and completion of courses so that SE's qualifications will be updated when they pass a new course.
    - ➢ As a supervisor I want to reserve seats in courses for SE's in my territory to increase their proficiency in different areas so I can better fill the needs of my territory.

- ➢ As a supervisor I want to update the skills of any SE currently serving under me so I can determine what I need to do to better to fill the needs of my territory.
- ➢ As a supervisor I want to add or remove employees from the list of those working under me so I know whether to hire people.
- ➢ As a product specialist I want create/update the courses needed to be approved to work on a piece of equipment.
- ➢ As a scheduler I want to view each SE's courses taken and skills so I can verify they have taken all the necessary courses.
- ➢ As a programmer I want to track various employee information, so I can maintain the overall list of employees.

Non-functional requirements:

1. Operational Requirements
    1.1 System can operate on any web browser
    1.2 System will automatically back up at midnight
2. Performance Requirements
    2.1 System must be available 24 hours a day, 365 days a year
    2.2 Response time between system and the user must be 4 seconds or less
    2.3 System will store and retrieve transactional information every three seconds.
3. Security requirements
    3.1 Information about customers issues limited to technicians and system operators
    3.2 Administrative personnel can view customer contact and billing information, but not issues to be fixed by technicians.
4. Cultural and political requirements
    4.1 The system must comply with regulations regarding customer privacy and the specifications needed by a technician to fix customer equipment.

Functional Requirements

1. Handle service orders.
    1.1 User logs request for service on website.
    1.2 Client sends request to server, which generates service order and puts it in the backlog.
    1.3 Server searches database for nearest Service Engineer with qualifications to fulfill service order that is not currently fulfilling other orders.
        1.3.1 If an service engineer is found schedule an appointment for them fix the user's equipment.
        1.3.2 If a service engineer is not found alert the supervisor within the user's territory that someone needs to be trained to fulfill the request, if a service engineer with the proper qualifications is not going to be free to fulfill the service order within one week.
    1.4 The service engineer fulfills the service order and logs that it has been completed.
    1.5 The service order is removed from the backlog.

2. Manage the Learning System

2.1 Schedulers update the list of available courses at periods of time.

2.2 System tracks registration in each course, and the number of open seats.

2.3 When a service engineer completes a course either the system or the instructor marks the course as complete.

2.4 Allow a supervisor update their skills.

3. Manage Employee Skills

3.1 Allow supervisors to reserve seats in courses for service engineers directly under them.

3.2 Allow supervisors to update the skills of each employee under them.

3.3 Allow supervisors to add or remove which employees work directly under them.

3.4 Allow Product Specialists to update the courses a service engineer needs to take to be proficient with a piece of equipment.

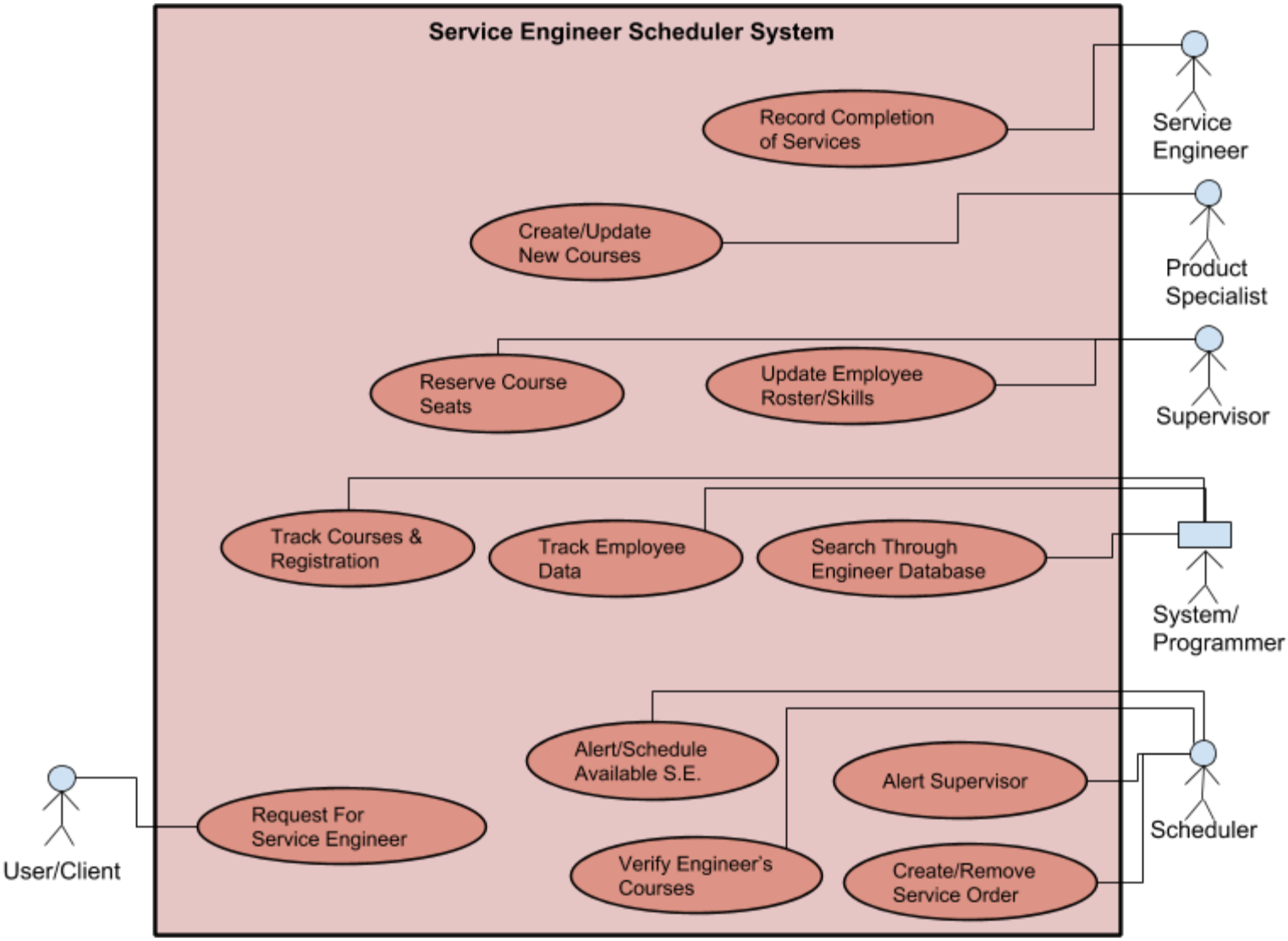
4. Manage Employee Records

4.1 Track when each employee was hired.

4.2 Track the status of each employee.

4.3 Retain employee records for auditing.

Functional model:



The above diagram illustrates 12 general use cases for the Scheduler System. This diagram involves 6 general roles including: User/Client, Scheduler, System/Programmer, Supervisor, Product Specialist, and Service Engineers.

Use-Case Descriptions:

Use Case	ID:1 Request for Service Engineer
Actors	User/Client
Stakeholders	User/Client - wants to schedule a service engineer to do specific work

	Scheduler - needs to know specific request to determine database query System - needs to know request for database query Supervisor - sees if additional training is needed for service engineer Product Spec. - determines if courses are suitable for order needed Service Eng. - responsible for working on service order by service request
Description	One of the first steps to finding an available engineer to work on request, made by the client or user
Trigger	User/Client has a need for an engineer and sends a service request to the system
Relationships	Association: User/Client
Flow of Events	1. Determine specific issue requiring a service engineer 2. Send formal request to system with specificies
Alternate flows	

Use Case	ID:2 Create/Remove Service Order
Actors	Scheduler
Stakeholders	User/Client - reads a copy of service order for validation Scheduler - allows internal/external users to review service order System - logs service order into system Supervisor - can view service order if anything else is needed Product Spec. - determines if courses are suitable Service Eng. - can view service order for reference
Description	Creation of a service order formally establishes an ongoing order that needs to be fulfilled.
Trigger	If Service Request received after Client sends service request, then create new order If Service Engineer records completion, then remove service order If No engineers available for service request, then remove service order
Relationships	Association: Scheduler, System
Flow of Events	1. Service Request received 2. Send formal request to system with specificies for work order
Alternate Flows	-Create Service Order -Delete Existing Service Order

Use Case	ID:3 Verify Engineer’s Courses
Actors	Scheduler
Stakeholders	Scheduler - will see if the engineer in the database has courses ready for work order

	System - searches through employee data and returns completed courses Supervisor - will find out if any engineers need to be trained Product Spec. - determines if courses are suitable
Description	Validates that the Engineer found in the database has the needed courses to be scheduled to a service order
Trigger	After engineer is found in database, scheduler verifies the engineer's courses are suitable for the requested work order
Relationships	Association: Scheduler, System, Product Specialist
Flow of Events	1. Service Request received 2. Engineer found in database 3. Compare service order with engineer's courses for validation
Alternate Flows	

Use Case	ID:4 Alert Supervisor
Actors	Scheduler
Stakeholders	Scheduler - notifies supervisor regarding engineers available for training Supervisor - can coordinate what engineers need to be trained
Description	Notifies supervisor according to what employees need to be trained
Trigger	If no suitable engineers for the job are found, alert supervisor to coordinate training
Relationships	Association: Scheduler, System, Supervisor, Service Engineer
Flow of Events	1. No suitable engineers found for service order 2. Proper supervisor is notified to see if an engineer can be trained within time.
Alternate Flows	

Use Case	ID:5 Alert/Schedule Available Service Engineer
Actors	Scheduler
Stakeholders	Scheduler - notifies and schedules engineer through the system System - tracks employee as well as updates schedule Service Engineer - notified of service order
Description	Alerts and schedules found Service Engineer to service order
Trigger	System finds a suitable engineer to work on service order

Relationships	Association: Scheduler, System, Supervisor Includes: Service Engineer
Flow of Events	1. System finds suitable engineer for work order 2. Service engineer is notified and scheduled to order after validating
Alternate Flows	

Use Case	ID:6 Search Through Engineer Database
Actors	System/Programmer
Stakeholders	Scheduler - creates query for search function based on service order System - searches through database for suitable engineer Service Eng. - provides data in database.
Description	Searches through engineer database and returns engineers that fits qualifications for order
Trigger	System finds a suitable engineer to work on service order based on specifications
Relationships	Association: Scheduler, System Includes: Service Engineer
Flow of Events	1. Scheduler sends system specifications for search query 2. System scans through database to find suitable engineer
Alternate Flows	

Use Case	ID:7 Track Employee Data
Actors	System/Programmer
Stakeholders	Supervisor - needs to know employee's completed courses System - perpetually updates employee's qualifications
Description	System tracks general employee data to database
Trigger	Employee updates location, qualifications New employee registers into system Supervisor requests for employee data
Relationships	Association: System Includes: Service Engineer, Supervisor,
Flow of Events	1. New employee registered into system 2. Existing employee updates information
Alternate Flows	- New employee registered

	- Existing engineer updates

Use Case	ID:8 Track Courses & Registration
Actors	System/Programmer
Stakeholders	Scheduler - needs to know if engineers have completed System - updates completed courses and registered courses from engineers Supervisor - views if engineers have completed courses needed for service order Product Spec. - creates courses and updates as needed Service Eng. - registers and completes courses recorded by system
Description	System tracks courses completed by engineers
Trigger	Engineer registers for course Product Specialist creates or updates course
Relationships	Association: System, Service Engineer, Product Specialist
Flow of Events	1. Employee registers into course 2. Product specialist creates new course or updates course
Alternate Flows	- Employee registers for a course - Product specialist creates new course or updates course

Use Case	ID:9 Update Employee Roster/Skills
Actors	Supervisor
Stakeholders	Scheduler - requires up to date roster to verify engineer for work order System - tracks employee roster, skills, Supervisor - views if engineers have completed courses needed for service order Product Spec. - creates courses and updates as needed Service Eng. - registers and completes courses recorded by system
Description	Supervisor tracks current employees and updates any new employees. If engineers complete a course or learn new skills then the supervisor tracks that as well
Trigger	New employee registered into system Employee learns new skill or takes new course
Relationships	Association: System, Supervisor Includes: Service Engineer
Flow of Events	1. Employee registers into course 2. Employee learns new skill or takes new course
Alternate Flows	- Employee registers for a course

	- Employee learns new skill or takes new course

Use Case	ID:10 Reserve Course Seats
Actors	Supervisor
Stakeholders	System - updates available course seats Supervisor - reserves course seats for engineers that need to be trained Product Spec. - creates courses and updates as needed Service Eng. - fills up course seats
Description	Supervisor reserves course seats for any engineers that require training in a specific area
Trigger	Service engineer needs to take a course within a week for a service order Supervisor requests reservation through system
Relationships	Association: System, Supervisor Includes: Service Engineer
Flow of Events	1. Engineer required to take course for training 2. Supervisor assigns course to engineer and reserves seats
Alternate Flows	

Use Case	ID:11 Create/Update Courses
Actors	Product Specialist
Stakeholders	System - records updates and new updates for courses Supervisor - can see available courses and course changes Product Spec. - creates courses and updates as needed Service Eng. - can view courses available to take
Description	Product Specialist can edit courses and update them as needed, while other users will know what courses have changed
Trigger	Course is not sufficient enough for a specific work order New course is needed as request by supervisor
Relationships	Association: System, Supervisor, Product Specialist Includes: Service Engineer
Flow of Events	1. Course not sufficient for specific work order 2. New course created to fill specifications 3. Course
Alternate Flows	

Use Case	ID:12 Record Completion of Services
Actors	Service Engineer
Stakeholders	System - establishes completion of existing order Client - contacted regarding completion of services Service Eng. - records when the service order is completed
Description	Upon completing the work order, the engineer notifies the system which in turn notifies the client.
Trigger	Service engineer completes order and notifies the system
Relationships	Association: System, Client
Flow of Events	1. Engineer completes work order 2. Engineer notifies system of completion
Alternate Flows	

Activity Diagram(s)

Functional Requirement 1: Handle service orders.

1.1 User logs request for service on website.

1.2 Client sends request to server, which generates service order and puts it in the backlog.

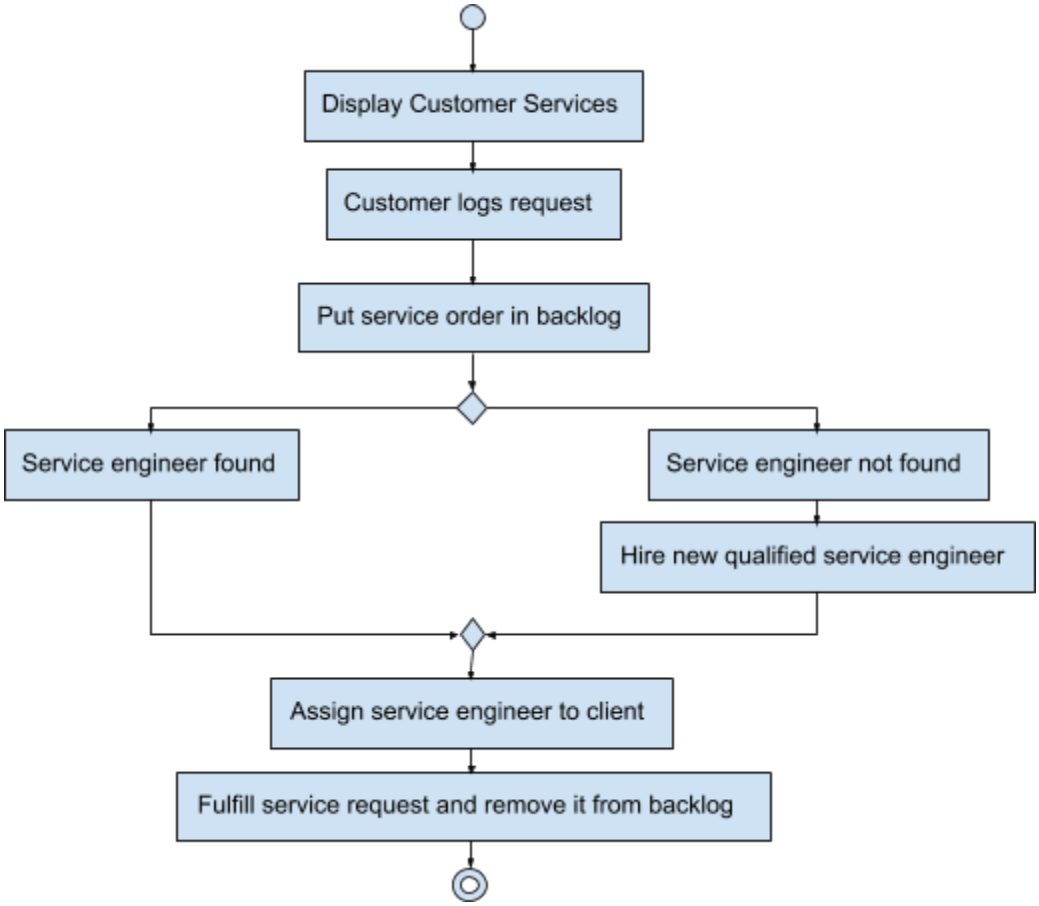
1.3 Server searches database for nearest Service Engineer with qualifications to fulfill service order that is not currently fulfilling other orders.

1.3.1 If an service engineer is found schedule an appointment for them fix the user's equipment.

1.3.2 If a service engineer is not found alert the supervisor within the user's territory that someone needs to be trained to fulfill the request, if a service engineer with the proper qualifications is not going to be free to fulfill the service order within one week.

1.4 The service engineer fulfills the service order and logs that it has been completed.

1.5 The service order is removed from the backlog.



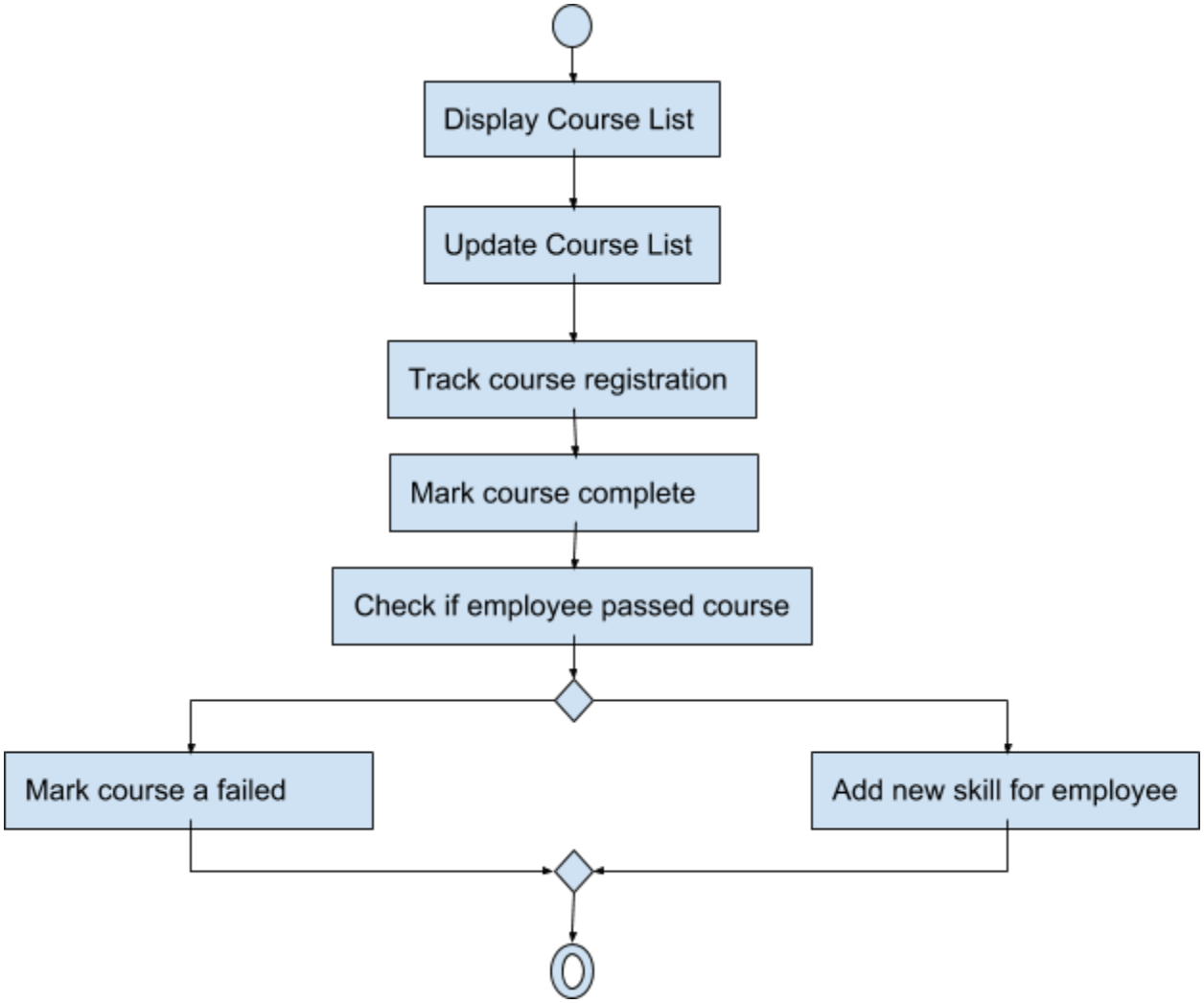
Functional Requirement 2: Manage the Learning System

2.1 Schedulers update the list of available courses at periods of time.

2.2 System tracks registration in each course, and the number of open seats.

2.3 When a service engineer completes a course either the system or the instructor marks the course as complete.

2.4 Allow a supervisor update their skills.



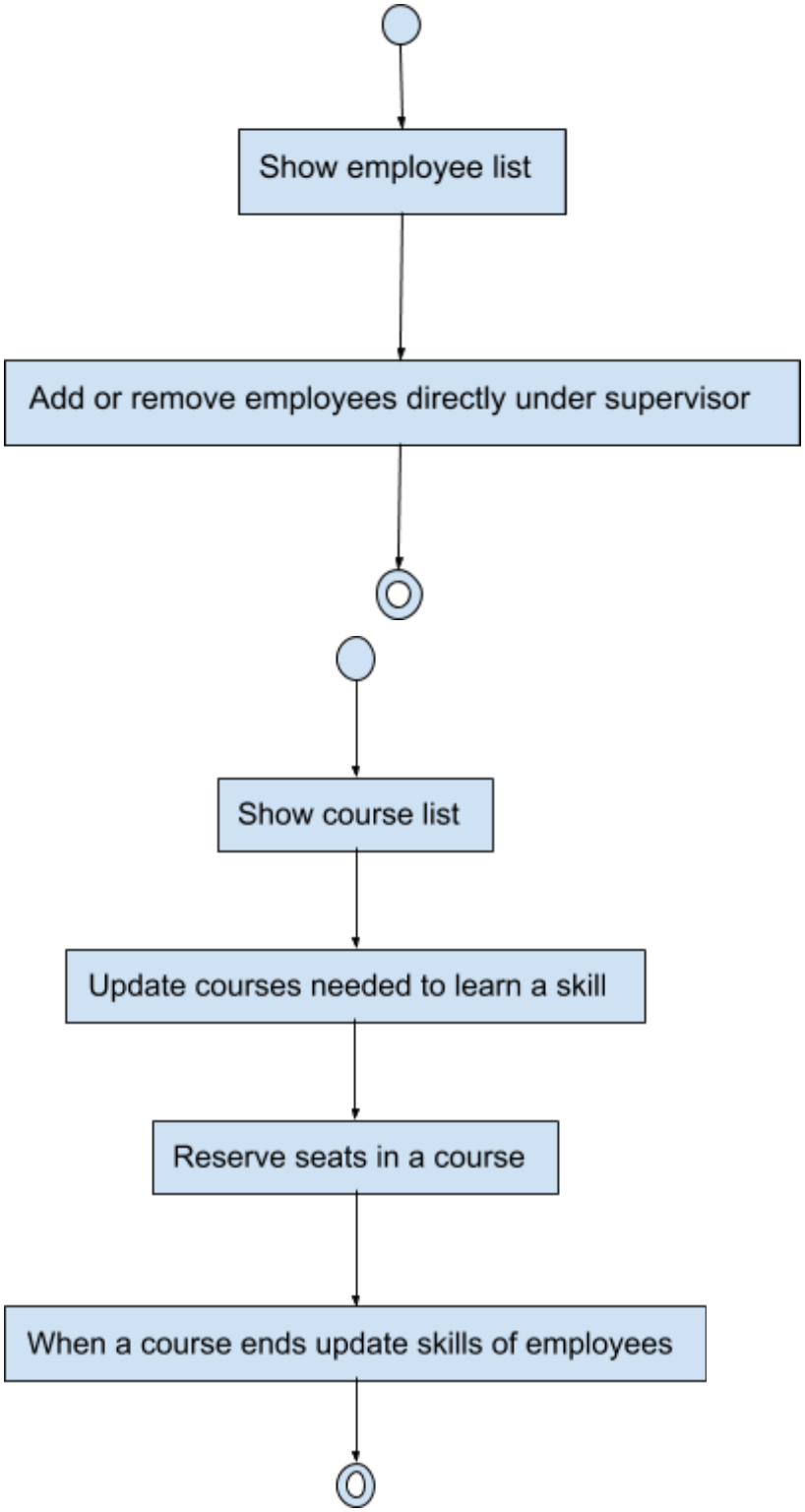
Functional Requirement 3: Manage Employee Skills

3.1 Allow supervisors to reserve seats in courses for service engineers directly under them.

3.2 Allow supervisors to update the skills of each employee under them.

3.3 Allow supervisors to add or remove which employees work directly under them.

3.4 Allow Product Specialists to update the courses a service engineer needs to take to be proficient with a piece of equipment.

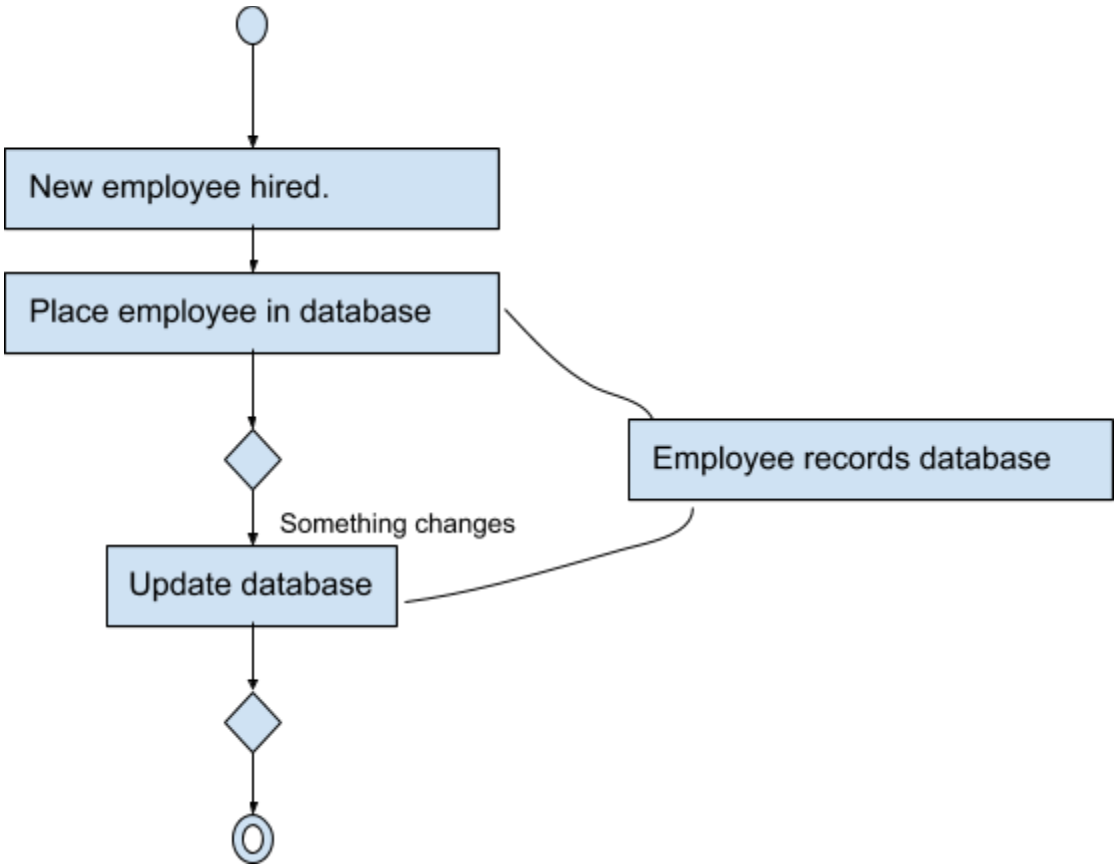


Functional Requirement 4: Manage Employee Records

4.1 Track when each employee was hired.

4.2 Track the status of each employee.

4.3 Retain employee records for auditing.


New employee hired.
Place employee in database
Employee records database
Something changes
Update database

Structural Model:

Here are our CRC cards for the system:

Front:

Class name:	ID:1	Type: Concrete, Domain
Description: Service engineers must complete specific courses before completing a service request		Associated use cases: 3, 4, 8, 10, 11
Responsibilities: Reserve course seats Create courses Update courses Verify courses Assign courses		Collaborators: Product Specialist Supervisor System/Programmer User

Back:

Attributes: Course status Eligibility Type
Relationships: Generalization: Program Aggregation: Employee Database Other associations: Employees

Front:

Class name: Customer service request	ID:2	Type: Concrete, Domain
Description: The customer requests to have a particular service completed for maintenance or repairs		Associated use cases: 1
Responsibilities: View request Display services needed Update status of request Find service engineer		Collaborators: Service engineer Supervisor

Back:

Attributes: Number of requests Completion time Severity
Relationships: Generalization: Call to action Aggregation: Customer service Other associations: Customer

Front:

Class name: Database	ID:3	Type: Concrete, Domain
Description: Databases provide information for the system to complete requests and for collaborators to work together on this platform		Associated use cases: 1,5,6,7
Responsibilities: Update Program Complete requests Respond to actions Store data		Collaborators: Service engineer Product specialist Supervisor System Scheduler

Back:

Attributes: Storage capabilities Information Speed Efficiency User interface
Relationships: Generalization: System Aggregation: All components of system Other associations: Employees

Front:

Class name: Service engineer	ID:4	Type: Concrete, Domain
Description: Person who takes courses and completes maintenance requests at customer sites		Associated use cases: 1,3,5,6,8,9,10,11

Responsibilities:	Collaborators:
Complete courses Go to customer sites Perform maintenance requests Log data into database Respond to any other customer requests	Service engineer Product specialist Supervisor System Scheduler Customer

Back:

Attributes:

Skills
Type
Availability

Relationships:

Generalization: Employee

Aggregation: All components of system

Other associations: System

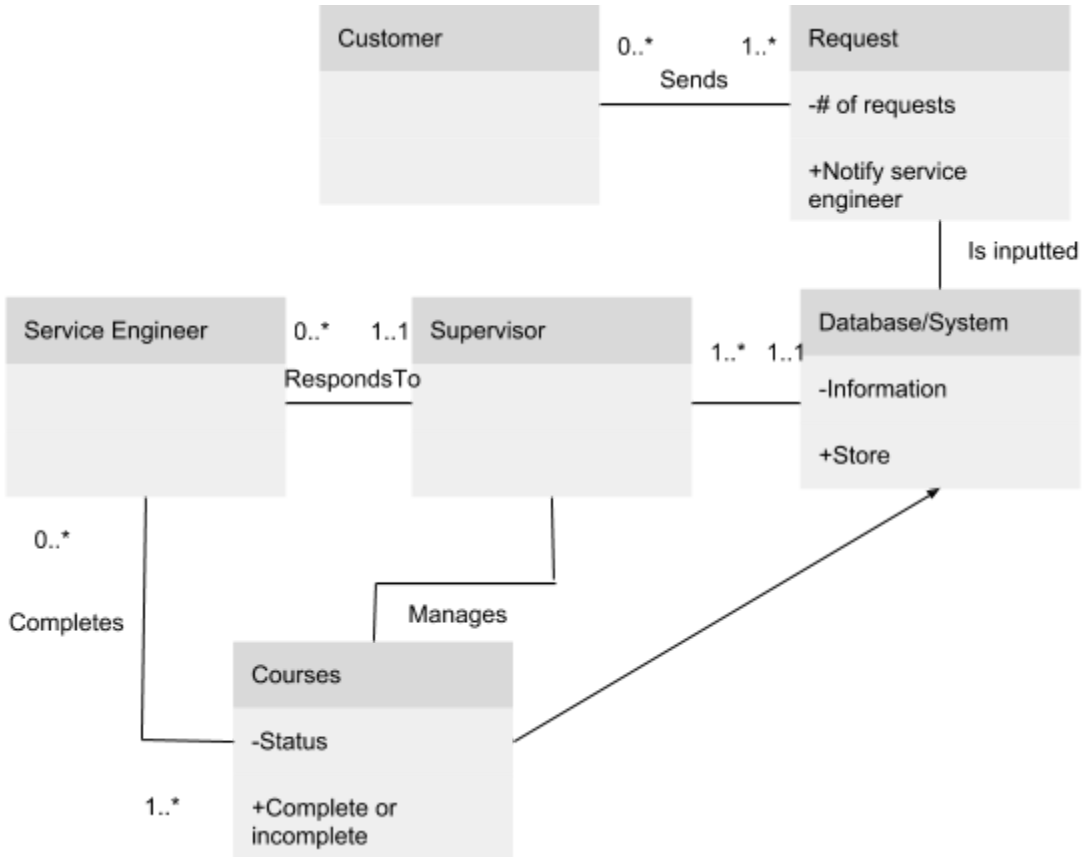
Front:

Class name: Supervisor	ID: 5	Type: Concrete, Domain
Description: The person who oversees service engineers and performs other duties		Associated use cases: 3, 5
Responsibilities: Reserve course seats Update employee roster Manage service engineers Oversee operations		Collaborators: Product specialist Service engineer

Back:

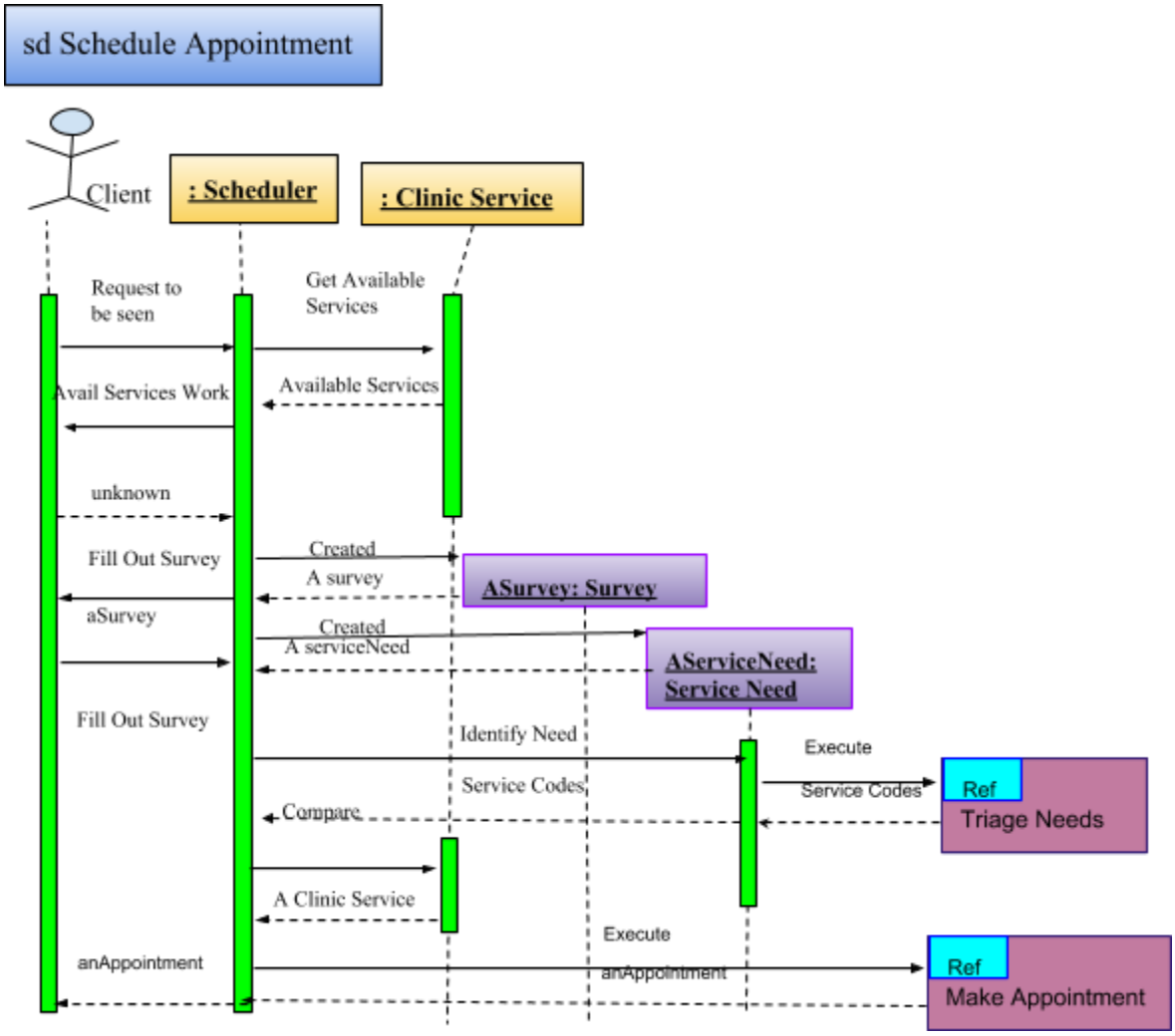
Attributes: Organized Efficient Manages well
Relationships: Generalization: Employee Aggregation: Database and courses Other associations: System

Class diagram for superclass:



Behavioral models:

★ Sequence Diagrams

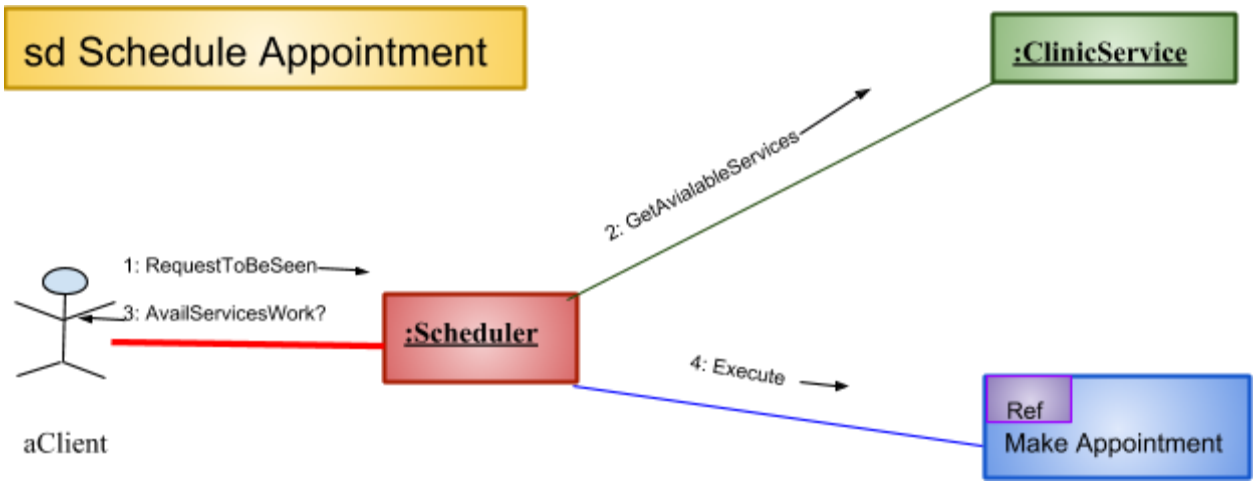


- **Sequence diagrams show:**
  - ➢ The messages that pass between objects over time.
  - ➢ The order of messages in an interaction.
- **The Setup:**

1. Determine the context of the diagram
2. Identify the actors and objects in the scenario being modeled.
3. Set a lifeline for each object.
4. To add the messages in order
5. The execution occurrence is added to each object and actor's lifeline.

- **The Process:**
  - Needed to be an intermediary between a client and the problem domain classes.
    - Non-mobile context; intermediary would be a receptionist.
    - Mobile context; the intermediary would be an actor-like class
      - Handle the interaction.
  - Add a Scheduler class
    - Requires modifications to the structural model
    - Schedule Appointment use case executes...
      - Make Appointment use case
      - Make Referral use case.
  - Client scenario
    - Schedule Appointment use case will call ...
      - Triage use case to make the determination as to the services require based on the survey.

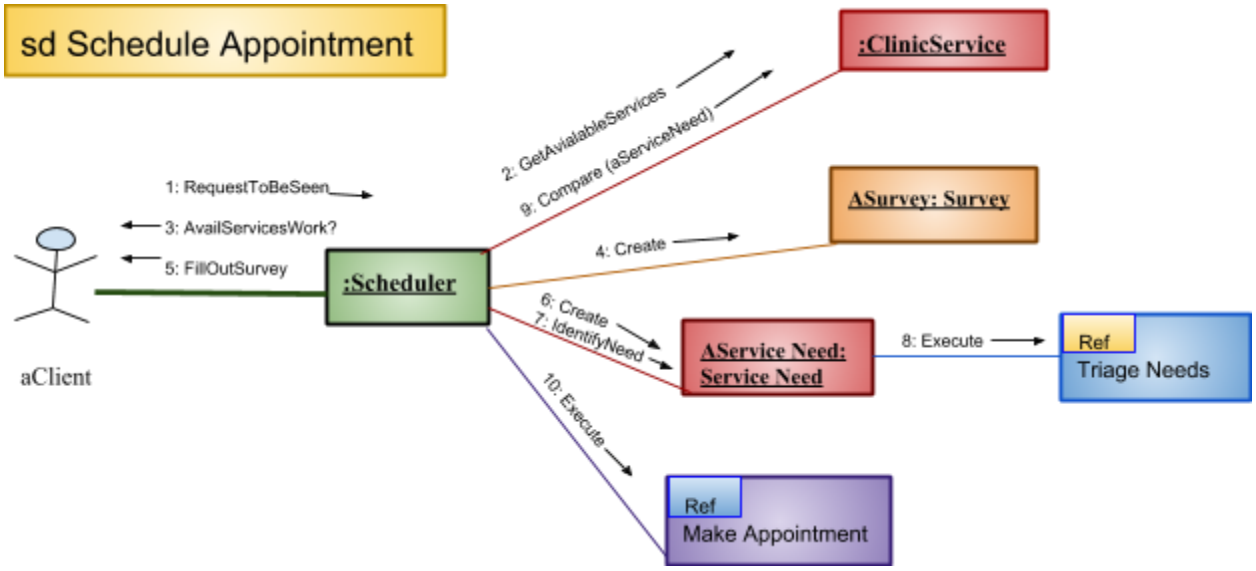
★ Communication Diagrams



- **Communication diagrams show:**
  - The messages that pass between objects.
    - Its valuable due to additional information that each diagram can uncover.
    - Emphasize the flow of messages across objects
- **Schedule Appointment use case:**
  - Utilizes the same classes used in the sequence diagram.

- Objects are positioned based on their associations with other objects in the use case and the messages are numbered.

★ Behavioral State Machines



- **Behavior diagrams show:**
  - Changes that occur within an object.
  - The transitions that an object passes through the execution of a use case.
- **Schedule Appointment use case:**
  - An instance of the client class, (aClient)
    - Transitions from searching, surveyed, and need defined
      - Either referred or to be scheduled.
    - Client class of the behavioral state machine
      - Uncover the need for status as an attribute for the Client class.

Crude Matrix for Employee Skills Management

	Supervisor	Service Engineer	Customer	Database/ System	Courses	Request
Supervisor		R, U, E		C,R,U,D		
Service Engineer				C,R,U,D		

Customer						C
Database/ System	C,R,U,D	C,R,U,D	C,R,U,D		C,R,U, D	
Courses	C	C	C			
Request						C,U

Method specifications:

To start off with method specifications, we will make contract forms to document each class in order to understand each method. Then, we will create method specifications for each method, using information from CRC cards such as class name as well as use cases. Contract forms and method specification forms are connected through ID number.

Method name: updateCourse	Class name: Course	ID: 1
Clients: course		
Associated use cases: updateEmployeeCourse		
Description of responsibilities: Allow course to be updated when employee completes the course.		
Arguments received: aCourse:Course		
Type of value returned: Success		
Pre-conditions: New course is not listed under current courses associated with employee		
Post-conditions: New course = old course + new order course		

Method name: updateCourse	Class name: Course	ID: 1
Contract ID: 1	Programmer: Product specialist	Date due: 4/10/2018
Programming language: Java		
Triggers/events: Employee takes course		
Arguments received: Data type: Course	Notes:	
Messages sent & arguments passed: ClassName.MethodName: -Course.UpdateCourse	Data type: Course	Notes:
Arguments returned: Data type: void	Notes:	

Method name: reserveCourse	Class name: Course	ID: 2
Clients: course		
Associated use cases: reserveEmployeeCourse		
Description of responsibilities: Reserve course seats for engineers needing to complete courses.		
Arguments received: aCourse:Course		
Type of value returned: Success		
Pre-conditions: New course is not listed as reserved for specific employee		
Post-conditions: New course = old course + reserved course		

Method name: reserveCourse	Class name: Course	ID: 2
Contract ID: 2	Programmer: Supervisor	Date due: 4/10/2018
Programming language: Java		
Triggers/events: Employee is required to take specific course.		
Arguments received: Data type: Course	Notes: Course is now reserved	
Messages sent & arguments passed: ClassName.MethodName: -Course.ReserveCourse	Data type: Course	Notes:
Arguments returned: Data type: void	Notes:	

Method name: requestServiceEngineer	Class name: Service Engineer	ID: 3
Clients: Customers		
Associated use cases: requestforServiceEngineer		
Description of responsibilities: Request a service engineer to work with a customer in completing maintenance.		
Arguments received: aRequest:Request		
Type of value returned: Success		
Pre-conditions: New request is not inputted for current customer		
Post-conditions: New request = no request + new request		

Method name: requestServiceEngineer	Class name: Service Engineer	ID: 3
Contract ID: 3	Programmer: Scheduler	Date due: 4/10/2018
Programming language: Java		
Triggers/events: Customer puts in request to have maintenance done.		
Arguments received: Data type: Service engineer	Notes:	
Messages sent & arguments passed: ClassName.MethodName: -ServiceEngineer.ReserveService Engineer	Data type: Course	Notes:
Arguments returned: Data type: void	Notes:	

Method name: recordServiceCompletion	Class name: Service Engineer	ID: 4
Clients: Service engineer		
Associated use cases: serviceCompletion		
Description of responsibilities: Once a service engineer completes a request, this will be recorded into the system.		
Arguments received: recordService:Service		
Type of value returned: Success		
Pre-conditions: New recording is not inputted for current request		
Post-conditions: New recording= no recording + new recording		

Method name: recordServiceCompletion	Class name: Service Engineer	ID: 4
Contract ID: 4	Programmer: Service engineer	Date due: 4/10/2018
Programming language: Java		
Triggers/events: A service is completed by a service engineer.		
Arguments received: Data type: Service Completion	Notes:	
Messages sent & arguments passed: ClassName.MethodName: -ServiceCompletion.RecordServiceCompletion	Data type: Service completion	Notes:
Arguments returned: Data type: void	Notes:	

we

Method name: trackCourse	Class name: Course	ID: 5
Clients: Service engineer		
Associated use cases: Track Courses & Registration		
Description of responsibilities: Track an engeneer's course completion and status of completion.		
Arguments received: trackCourse:Course		
Type of value returned: Success		
Pre-conditions: void		
Post-conditions: void		

Method name: trackCourse	Class name: Course	ID: 5
Contract ID: 5	Programmer: System/Database	Date due: 4/10/2018
Programming language: Java		
Triggers/events: A service engineer is completing the course which prompts the system to keep track of their progress.		
Arguments received: Data type: Course	Notes:	
Messages sent & arguments passed: ClassName.MethodName: -Course.TrackCourse	Data type: ourse	Notes:
Arguments returned: Data type: void	Notes:	

Method name: searchDatabase	Class name: Database	ID: 6
Clients: Service engineer, supervisor		
Associated use cases: Search through engineer database		
Description of responsibilities: The system can search through the engineer database to find appropriate information regarding the specific service engineer and which courses they have completed or are completing.		
Arguments received: searchDatabase:Database		
Type of value returned: Success		
Pre-conditions: Locate information		
Post-conditions: Obtain new information		

Method name: searchDatabase	Class name: Database	ID: 6
Contract ID: 6	Programmer: System/Database programmer	Date due: 4/10/2018
Programming language: Java		
Triggers/events: When an employee wants to retrieve information about a service engineer and their current status with courses or work performed.		
Arguments received: Data type: Database	Notes:	
Messages sent & arguments passed: ClassName.MethodName: -Database.SearchDatabase	Data type: Database	Notes:
Arguments returned: Data type: void	Notes:	

Method name: scheduleServiceEngineer	Class name: Service Engineer	ID: 7
Clients: Service engineer		
Associated use cases: Schedule service engineer		
Description of responsibilities: Alert the service engineer and schedule them based on customer requests as well as service engineer's availability.		
Arguments received: scheduleServiceEngineer:ServiceEngineer		
Type of value returned: Success		
Pre-conditions: Locate service engineer availability		
Post-conditions: Enter scheduling information		

Method name: scheduleServiceEngineer	Class name: Service Engineer	ID: 7
Contract ID: 7	Programmer: Scheduler	Date due: 4/10/2018
Programming language: Java		
Triggers/events: When a request is approved and a service engineer needs to be added to the schedule, they are scheduled into the system's calendar.		
Arguments received: Data type: Service engineer	Notes:	
Messages sent & arguments passed: ClassName.MethodName: -ServiceEngineer.ScheduleServiceEngineer	Data type: Service engineer	Notes:
Arguments returned: Data type: void	Notes:	

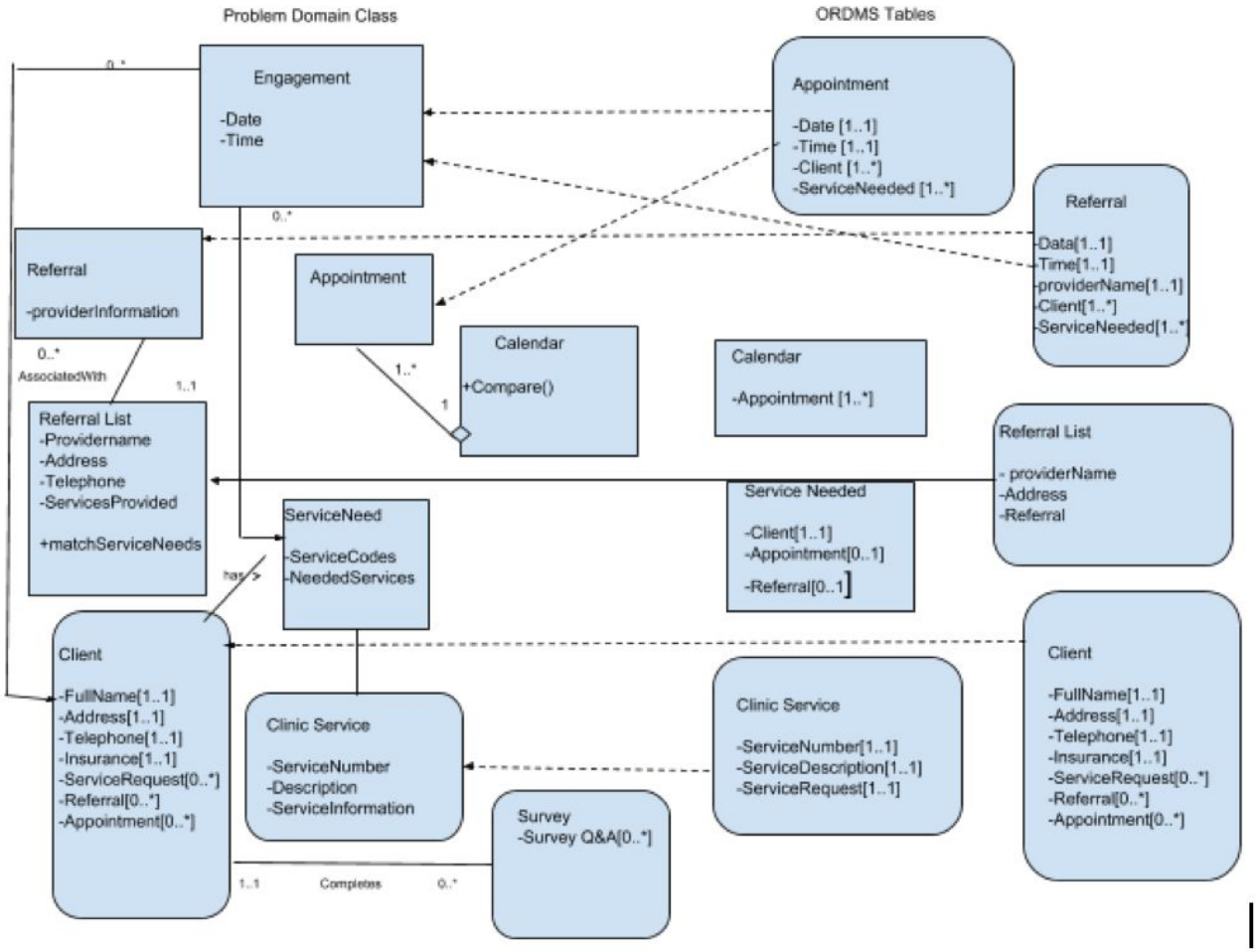
Method name: alertSupervisor	Class name: Supervisor	ID: 8
Clients: Supervisor		
Associated use cases: Alert supervisor		
Description of responsibilities: To alert the supervisor when a service engineer is scheduled for a repair or when they have completed their courses.		
Arguments received: alertSupervisor:Supervisor		
Type of value returned: Success		
Pre-conditions: void		
Post-conditions: Send alert to supervisor		

Method name: alertSupervisor	Class name: Supervisor	ID: 8
Contract ID: 8	Programmer: Scheduler	Date due: 4/10/2018
Programming language: Java		
Triggers/events: Send an alert to the supervisor if service engineer needs to complete course or complete job and when they have completed them.		
Arguments received: Data type: Supervisor	Notes:	
Messages sent & arguments passed: ClassName.MethodName: -Supervisor.alertSupervisor	Data type: Supervisor	Notes:
Arguments returned: Data type: void	Notes:	

Method name: createServiceOrder	Class name: Service order	ID: 9
Clients: Service engineer, Supervisor		
Associated use cases: Create or update a service order		
Description of responsibilities: To create a service order for a service engineer to go out into the field which responds to a customer request.		
Arguments received: alertSupervisor:Supervisor		
Type of value returned: Success		
Pre-conditions: No service order initiated		
Post-conditions: New service order		

Method name: createServiceOrder	Class name: Service order	ID: 9
Contract ID: 9	Programmer: Scheduler	Date due: 4/10/2018
Programming language: Java		
Triggers/events: Customer request is received.		
Arguments received: Data type: Service order	Notes:	
Messages sent & arguments passed: ClassName.MethodName: -ServiceOrder.createServiceOrder	Data type: Service Order	Notes:
Arguments returned: Data type: void	Notes:	

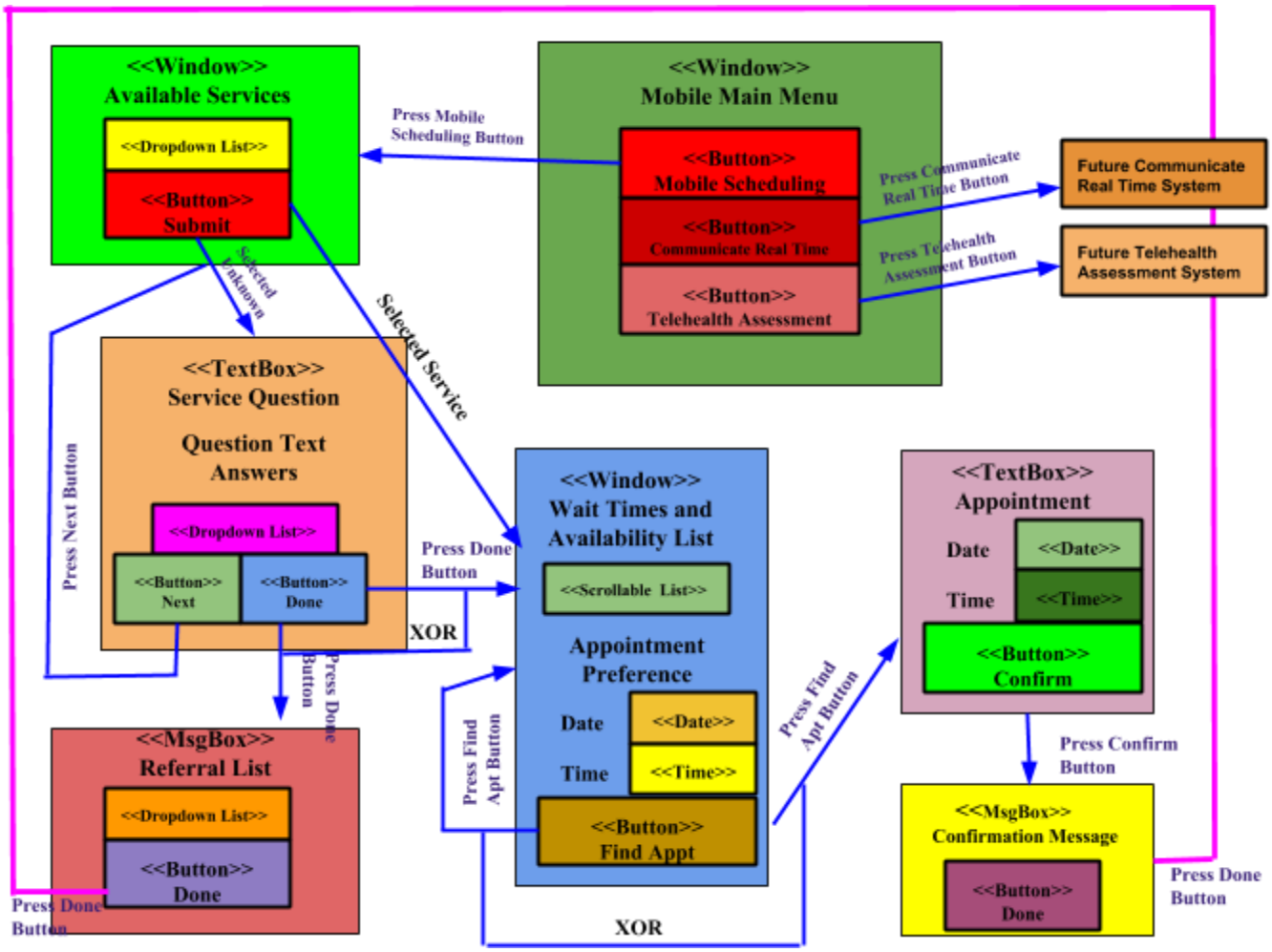
Problem Domain Class ORDBMS table mappings



Schema for Mapping Problem Domain Objects to ORDBMS

We have mapped all the concrete Problem Domain classes to the ORDBMS tables. The single valued attributes to the columns of the ORDBMS have also been mapped. The mapping of the single-valued aggregation and association relationships that store the Object ID have been fulfilled. The repeating groups of attributes to a new table and the creation of the one-to-many association from the original table has been made to the new one.

Navigation Structure Diagram

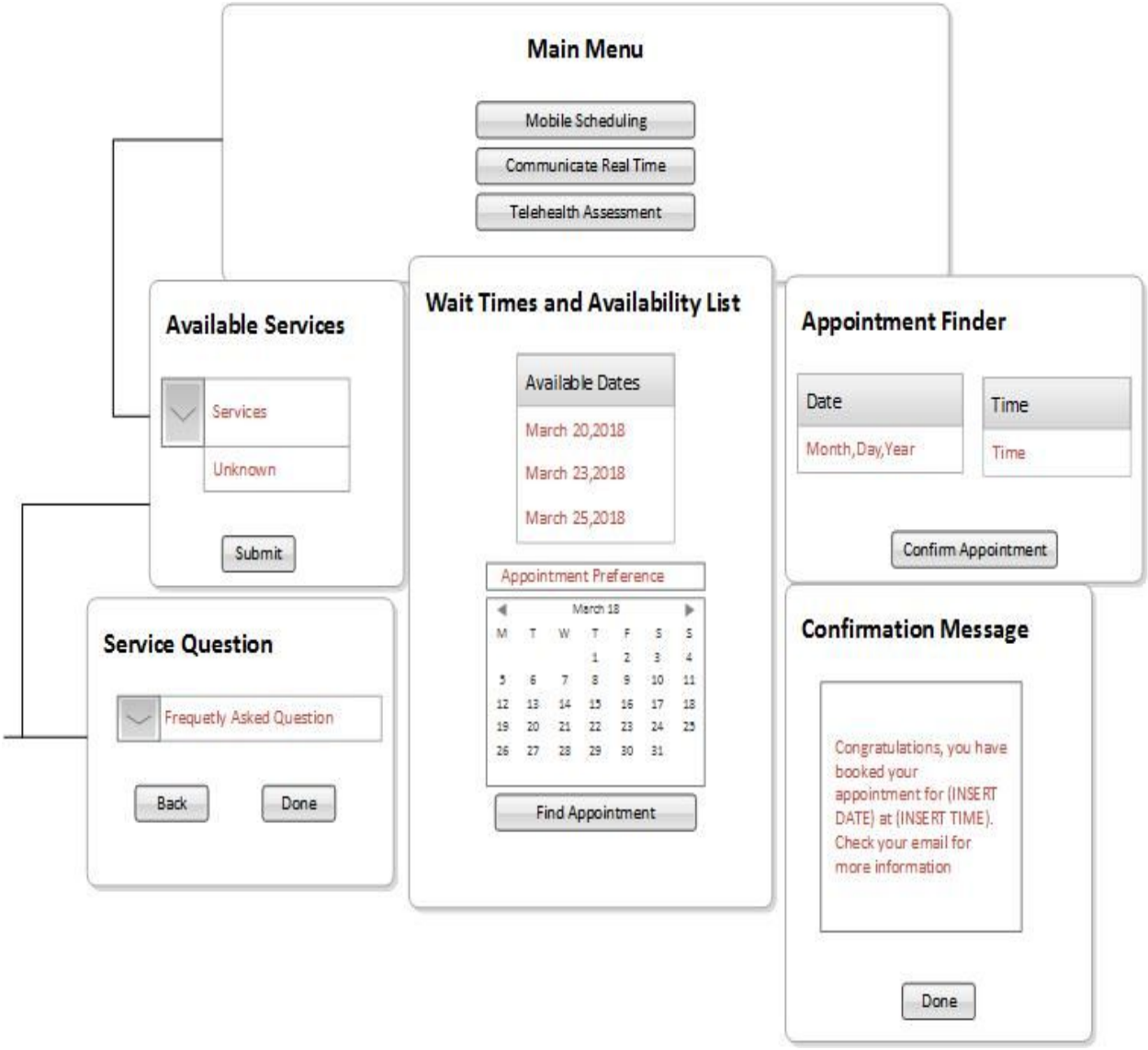


Designed the structure of the navigation through the different user interface components that would be required to support the user in scheduling an appointment. We created a WND that portrays all of the differing paths through the user interface. This WND only documents the navigation through the user interface components that support the Mobile Scheduling aspect of the system.

How the diagram works:

- The first use begins with pressing the Mobile Scheduling Button at the top center of the diagram. Next, the client selected a service from the dropdown list and presses the Submit button to move onto the Wait Times and Availability List window where the wait times are displayed in a scrollable list. The client can enter the preferred date and time for the appointment being scheduled by pressing the Find Appt button, which requests the system to create an appointment on the preferred date and time. The XOR is connecting the two Press Find Appt button paths. What this does is that only one of the paths will be processed at a time. If the system does not find the correct date and time, the system will ask the client to try again. Lastly, the system will travel to the other path and ask the client confirm the appointment by clicking the Confirm button. The system will send a confirmation message back to the client. Finally, the Client presses the Done button to return to the Main Menu of the system.
- The second use begins with pressing the Mobile Scheduling Button again. Then the client selects unknown and presses the Submit button to move onto the Survey interface component. Next, the client will answer each question by selecting an answer from the dropdown list with each question. The client can either go to the next question, (clicking Next button), or will go on to the Wait Times and Availability List window (Clicking the Done button). The Done button will be available until the entire set of questions have been answered. Again the XOR arc, (works the Done button) which is processed on the system determining whether the clinic did provide the relevant service.

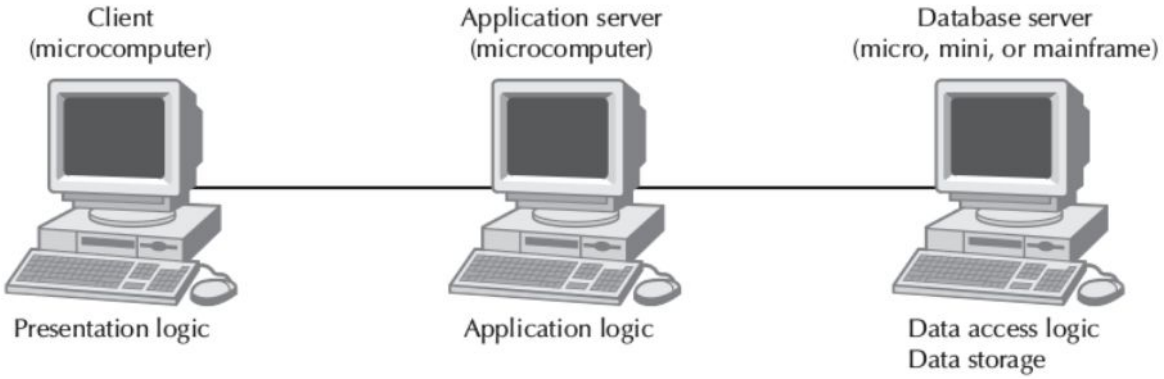
Interface Design Prototype



Navigation Model

Step 1: Determine what architecture the system we will use.

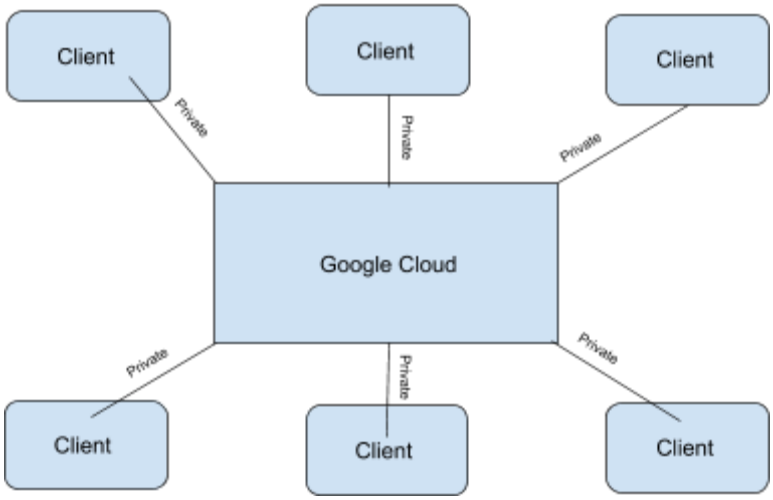
We will use a three tiered client-server architecture. The clients will be end-user devices like computers and smartphones and will handle presentation logic. Application servers will handle the application logic and database servers will handle the data access logic and data storage. Assuming all tiers are computers of some sort, the architecture will be similar to:



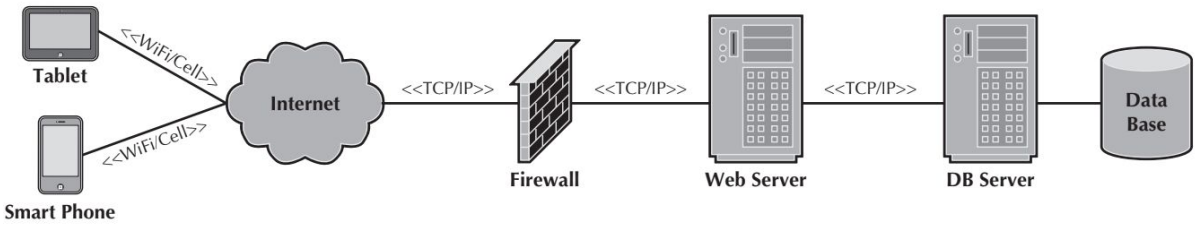
While using a three tiered client-server architecture will incur higher development costs, they will be offset by the low cost of infrastructure, ease of development, and scalability of the system. The clients will developed as web-apps, the application server will be handled by the Google Cloud, and the database server will be managed and provided by Google through the Google Cloud.

Step 2: Create the network model of the system.

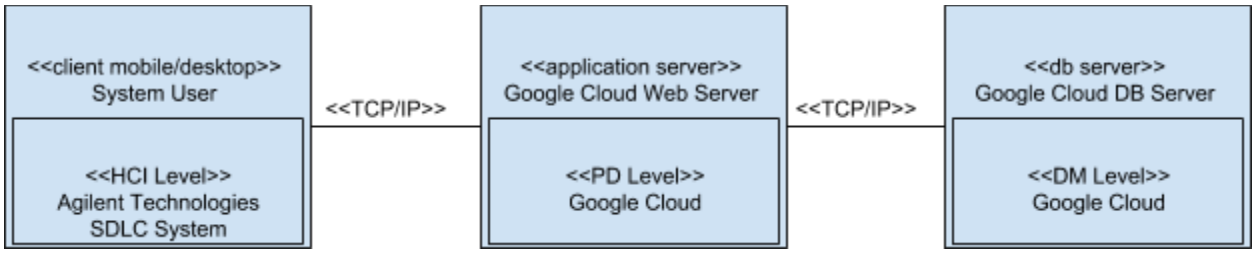
Because we are using the Google Cloud for our application and database servers, we don't have to maintain regional databases. The clients connect into the Google Cloud to enter requests. The network model is:



Given more description, the diagram for any client is:



The client device connects into the web server using the internet, which retrieves data from the db server. The web server performs the application logic for the system and sends the data back to the client device which presents it to the user. Showing the data transfer between different layers of the system, the diagram is:



The client device handles the human computer interaction for the system, taking inputs from the user and sending them to the application server managed by the Google Cloud. The application server represents the problem domain layer of the system, handling the application logic for the system. The application server requests data from the db server which is also run by the Google Cloud when it needs it to perform various calculations. The db server handles the data management functions for the system, including the data access logic and data management. Data is transferred between layers using TCP/IP protocols.

Hardware and Software Specs

Because we are using the Google Cloud to handle the the application and database servers, we will not create specs for them. That is up to Google. We will create the hardware and software specs for the clients we will be developing.

Step 1: Define the software that will run on each component.

We would create a web application so the client needs to run an operating system and a web browser for our website to work. Potential combinations are:

Operating System	Web Browser
Windows	Internet Explorer
Mac OS X	Safari

Step 2: Create a list of the hardware that is needed to support the future system.

The hardware will be created by the clients manufacturer but some sample hardware specifications would be:

Hardware	Network
Memory	Ethernet
Microprocessor	WiFi

We will not specify the hardware capacity for the clients since we will not be creating them, just using them to run our web app.

Step 3: Create the final specification for a sample client.

Specification	Sample Client
Operating System	Windows or Mac OS X
Special Software	Internet Explorer or Safari
Hardware	Memory Discs and Microprocessors
Network	Ethernet or WiFi

Nonfunctional Requirements

Step 1: Determine the operational requirements for the system.

Technical Environmental Requirements	- The system will work with any web

	browser. - All office locations will have an always-on network connection to enable real-time database updates.
System Integration Requirements	- The system will read and write to the main inventory database in the inventory system.
Portability Requirements	- The system must be able to work with different operating systems (e.g., Linux, Mac OS, and Windows). - The system must work on handheld device.
Maintainability Requirements	- New versions of the system will be released every six months.

Step 2: Determine the Performance Requirements for the system.

Speed Requirements	- Response time must be less than 7 seconds for any transaction over the network. - The databases must be updated in real time.
Capacity Requirements	- There will be a maximum of 150 users at any time.
Availability and Reliability Requirements	- Scheduled maintenance shall not exceed one 6-hour period each month. - The system shall have 99% uptime performance.

Step 3: Determine the Security Requirements for the system.

Access Control Requirements	- Only allow instructors to mark if a course is complete. - Only allow Supervisors to update the skills an employee has. - Only allow Product Specialists to update the requirement list to earn a specification. - Allow Admins to perform any task.
Encryption and Authentication Requirements	- Data will be encrypted from the user's computer to website to provide secure use of the system.

	- Users logging in from outside the office will be required to authenticate.
Virus Control Requirements	- All uploaded files will be checked for viruses before being saved in the system.

Step 4: Determine the Cultural and Political Requirements for the system.

Customization Requirements	- Allow Product Specialists to update the requirement list to earn a specification. - Allow Supervisors to update the skills an employee has. - Allow instructors to mark if a course is complete.
Legal Requirements	- Personal information about customers cannot be transferred out of European Union countries into the United States. - It is against U.S. federal law to divulge information on who rented what videotape, so access to a customer's rental history is permitted only to regional managers.