

Software Engineering Department  
ORT Braude College

Capstone Project Phase A – 61998

**Shelter Construction Management**

**24-1-D-43**

Project Supervisor: Assoc. Prof. Zeev Barzily.

Project Members:

Almog Khaikin, [Almog.Khaikin@e.braude.ac.il](mailto:Almog.Khaikin@e.braude.ac.il).

Nir Betesh, [Nir.Betesh@e.braude.ac.il](mailto:Nir.Betesh@e.braude.ac.il).

**Table of Contents**

Abstract 1

Introduction 2

Background and related work 3

Expected achievements 5

Engineering process 6

Research 6

Use Case Diagram 8

Class Diagram 10

Activity Flow Chart 11

Requirements 13

Software architecture overview 14

Evaluation/Testing 16

Bibliography 19

**Abstract**

This project aims to develop a web-based software solution specifically designed for a shelter construction company, which has historically managed its operations through cumbersome and error-prone paper-based methods. The primary goal is to transition the company's workflow into a digital environment, thereby reducing the need for excessive manual labor, minimizing errors, and enhancing the ability to manage multiple projects simultaneously.

Current generic solutions, such as Enterprise Resource Planning (ERP) systems, often fail to meet the unique needs of the company due to its complexity and steep learning curves, particularly for employees with limited technological expertise. Our software addresses this gap by providing a simplified, intuitive interface tailored to the specific workflows and roles within the company, such as secretaries, installers, and inventory managers.

This system will centralize and streamline the management of contracts, client interactions, progress tracking, and payments. By leveraging cloud technologies, the solution will offer scalable computing resources, reliable data storage, and easy migration capabilities, enhancing overall efficiency and security. Additionally, the system's design facilitates quick and effective training for employees, aligning with the company's operational needs and enhancing productivity without the technical overhead typically associated with traditional ERP systems.

The success of this project will be measured by the system's usability, performance under load, and its ability to significantly improve the management processes compared to the current manual system. Through this project, we expect to demonstrate a model that could be replicated or adapted for similar industries facing technological integration challenges.

1. **Introduction**

A shelter construction company has been managing all their work through paperwork. This workflow is error-prone, tedious and involves a lot of manual effort. Having to manage complex processes inside the company using paper requires the company to hire additional staff to do all the manual work, limit the number of simultaneous customers that they service and, in turn, miss out on potential profit.

Although there are many existing solutions that help manage these kinds of company related work (ERP, etc.), most of these solutions are designed to be quite generic to serve many different purposes. As a result, these solutions can be complex and difficult to learn how to operate. The company in question has many office workers who are not technologically proficient, and thus need a simpler solution that is tailored to the needs of the company.

The main stakeholders of this project are the company office workers who will be using the software to manage the company. The company’s clients will not be interacting directly with the software but rather the software will be used by company secretaries on behalf of the company’s clients.

We propose to develop a web-based dedicated software solution that will assist in managing all the company’s processes. This software will help manage contracts, clients, tracking of construction progress, and payments. By creating a dedicated solution, we can keep the complexity of the system to a minimum and speed up employee training. The software will be tailored to the needs of the company by including menus for each company-related activity that each relevant employee can access whenever they need to.

1. **Background and related work**
   1. Enterprise Resource Planning (ERP)[1]
      1. Historical Development of Enterprise Resource Planning

* ERP systems originated in the 1960s and 1970s with the development of Material Requirements Planning MRP systems focused on manufacturing processes.
* The concept evolved into Manufacturing Resource Planning(MRPII) systems in the 1980s, incorporating additional functionalities like finance, human resources, and project management.
* By the 1990s, ERP systems emerged as integrated suites of software modules covering various business functions, such as SAP's R/3, Oracle Financials, and Baan.
  + 1. Approaches and Systems
* Traditional On-Premises ERP Systems: Historically, most ERP systems were installed and maintained on-premises, requiring significant investment in hardware, software, and IT infrastructure.
* Cloud-Based ERP Systems: With advancements in cloud computing, cloud-based ERP solutions have gained popularity. Vendors like Oracle, SAP, and Microsoft offer cloud-based ERP platforms, providing scalability, flexibility, and reduced upfront costs.
* Open-Source ERP Systems: Open-source ERP solutions like Odoo, ERPNext, and Apache OFBiz offer customizable and cost-effective alternatives to proprietary ERP systems, suitable for small to medium-sized enterprises (SMEs).
* Industry-Specific ERP Systems: Some ERP vendors specialize in providing industry-specific solutions tailored to the unique requirements of particular sectors, such as manufacturing, healthcare, retail, and services.
  + 1. Technologies and Engineering Approaches
* Integration Technologies: ERP systems rely on integration technologies like Application Programming Interfaces (APIs), middleware, and enterprise service buses (ESBs) to connect disparate systems and facilitate data exchange.
* Database Management Systems (DBMS): Most ERP systems utilize relational database management systems (RDBMS) like Oracle Database, Microsoft SQL Server, or PostgreSQL for storing and managing data.
* Mobile ERP: The proliferation of mobile devices has led to the development of mobile ERP applications, allowing users to access ERP functionalities on smartphones and tablets.
* Artificial Intelligence (AI) and Analytics: Modern ERP systems leverage AI, machine learning, and advanced analytics to provide predictive insights, optimize processes, and support decision-making.
  + 1. Application Possibilities
* Customization and Configuration: ERP systems offer extensive customization and configuration options to adapt to the specific needs and workflows of different industries and organizations.
* Business Process Reengineering (BPR): Implementing an ERP system often involves reengineering existing business processes to align with best practices and optimize efficiency.
* Change Management: Successful ERP implementation requires effective change management strategies to ensure user adoption, minimize resistance, and maximize ROI.
* Continuous Improvement: ERP systems support continuous improvement initiatives by enabling data-driven decision-making, performance monitoring, and feedback mechanisms.
* By understanding the existing landscape of ERP systems, approaches, and technologies, organizations can make informed decisions when selecting, implementing, and optimizing their ERP solutions to meet their business needs effectively.

1. **Expected achievements**

We expect to develop a complete software solution for managing a company that constructs shelters for clients on a contract basis. The software will provide an easy to use, streamlined way to create contracts, assign teams to contracts, track the progress of construction work, manage company inventory, track payments and store the history of company work.

The software will use a database to hold all the relevant information about clients, contracts and the inventory. The database, along with the backend of the software system, will reside in the cloud which will enable cheap hosting, scalable computing resources and easy migration from place to place.

The use of a dedicated solution combined with the use of the cloud will ideally enable both easy installation of the system and fast training of employees in how to use the system. This is important since most of the company’s employees are not technically inclined, and so they require any help they can get to make the use of the system as simple and frictionless as possible.

The main software client will be built using web-based technologies which will enable rapid development and iteration and make the client usable from any device at any location.

Our criteria for success are to have a functioning construction company management system that is easy to use and allows managing the company in a manner that is substantially better than the current method the company is using. The user interface should be clear and easy to follow. The system should perform well without any hitching or stuttering while the software is processing work and allow multiple employees to access the system simultaneously without any conflicts or race conditions. Finally, the system should concisely present all the information the company needs to run their business.

We expect our project to have several unique features:

The system will take advantage of cloud infrastructure to ensure maximal availability of the system, improved scalability, good security for company information and automated backups of the system’s database in case of failures. The use of the cloud will enable quick iteration on the system since cloud services make it incredibly easy to deploy to them and it will also allow for quick and easy installation of the system on the company’s devices.

The system will also utilize a web-based user interface. This will enable the use of the software from almost any device as long as it has a web browser. The web-based interface will be built using modern full-stack frameworks that allow developers to easily mix backend and frontend code in the same codebase. Most of the work will be done in the frontend with the backend mainly used to fetch or update data in the database.

1. **Engineering process**
   1. Research

During the first semester, we focused mostly on understanding the requirements of the project and modeling how a solution to the problem would look like. We chose to rely primarily on researching the problem through conversations with people that are familiar with the subject matter. We chose this approach because we can learn a lot more about the subject through free conversations with people where we can ask any question on demand and get an immediate answer without having to waste a lot of time searching for information online and sifting through unimportant or misleading information.

Our primary source of information was the CEO of the shelter construction company whose company’s needs have been the main driving force behind the development of this project. We had many extensive meetings with him during which we discussed what the problem he was trying to solve was and what an ideal solution would be for him and his company. The company CEO did a lot of homework on our behalf, preparing various documents describing how his company operates as well as a detailed description of how he wants our software solution to work. He then presented these documents to us during our meetings where we could ask questions, clarify additional points and expand our knowledge on the problem domain and the desired solution.

Through our conversations with the company’s CEO, we learned various important details such as:

* 1. Product
* What types of employee roles exist in the company and how each of them is supposed to interact with our software: Secretary, team leader, inventory manager, CEO and construction workers.
* What are the steps in the process of the service the company provides: Contract signing, approval of permits, construction of door and window frames, installation of windows and doors and contract completion.

A second major source of information was our project supervisor who we later learned worked in construction in the past. We had many meetings with him throughout the semester in which he told us about how construction of buildings works and specifically how construction of shelters works during that process. His advice and information helped guide us on how to structure the workflow of our software solution and understand the set of features that we will need to implement.

Additionally, we looked at various existing software solutions that try to solve similar problems, tried to learn how they work, how they are structured, what features do they have, and we attempted to adapt the ideas from these solutions to our more specific problem. There are many ERP software solutions out there that serve many purposes like our own and we figured it would be best not to attempt to reinvent the wheel but rather learn from the previous efforts of large enterprise organizations. We took inspiration from how their UI was designed, how much automation vs manual work they required, and we also examined the kinds of places where these existing solutions were used in.

As we are gearing up for the implementation phase of our project, we intend in the future to start researching more practical topics such as the technologies we intend to use, the programming language, frameworks, cloud hosting solutions, databases and more. We will evaluate the many options we have for choice of technology based on various criteria such as ease of use, flexibility, performance and portability.

There are several constraints on the solution we intend to develop that we have to take into consideration:

* Most of the company’s employees are non-technical. This means the software must be simple to learn and use.
* The software will manage important company information about clients and contracts, so the data must be stored reliably and securely. This means using secure cloud storage solutions with automated backups in case of failures.
  + 1. Use case diagram

The following diagram describes the types of interactions users can have with the system as well as who can perform which interaction:

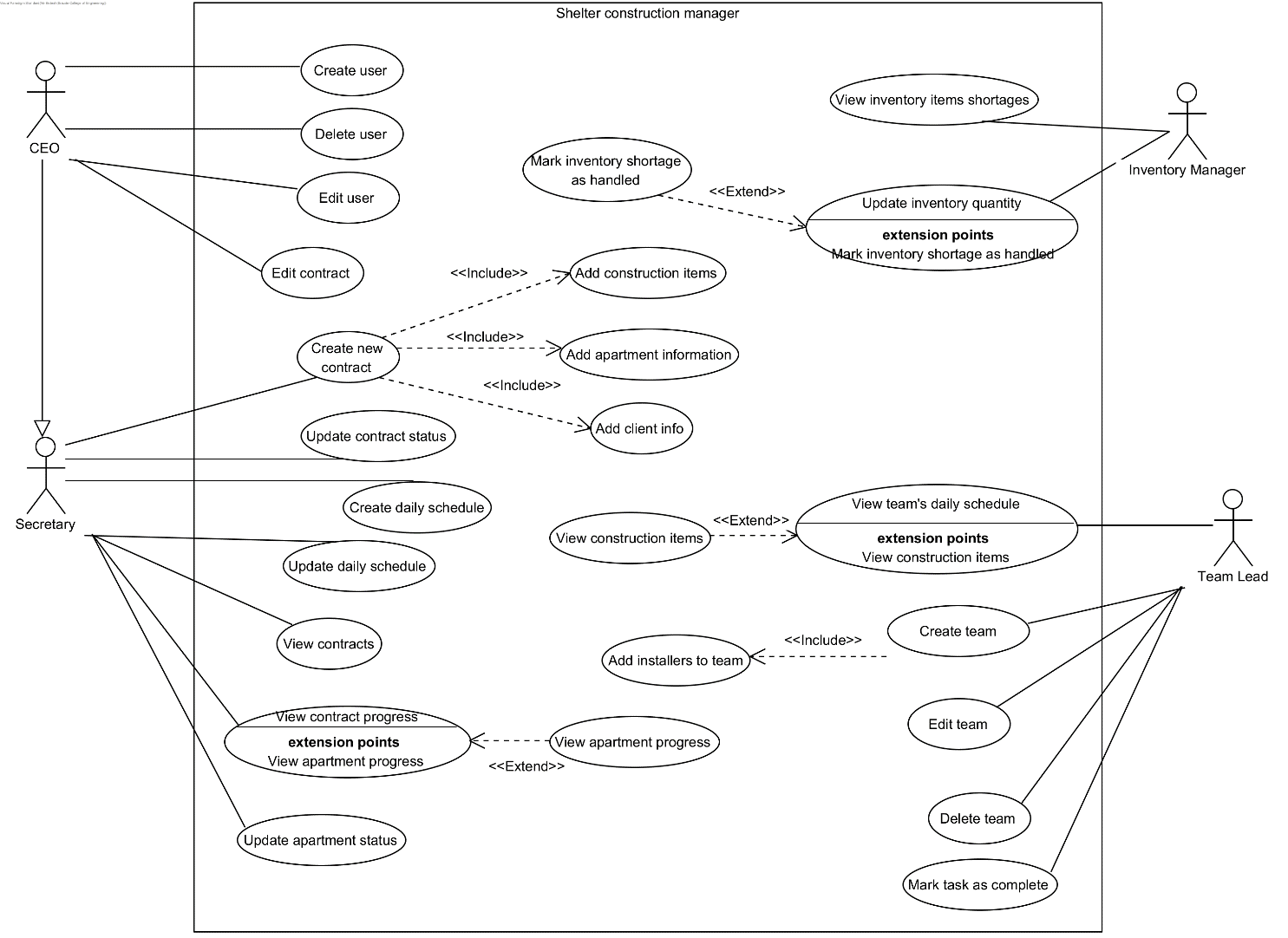


Figure 1: Use case diagram

The secretaries communicate with customers to sign contracts. Once a contract has been signed, the secretary that accepted the contract is the one assigned to the contract/customer. The secretary manages all aspects of the contract: The sending/receiving/filing of permits, ordering of materials, preparing the schedules of the installers that are assigned to their contracts, and marking contracts as completed. When the contract is initially created, it is in the pending state. When the customer accepts the contract, the secretary marks the contract as active (no longer pending). The secretary can also edit the contract after it was created.

The secretaries and CEO can view what contracts exist in the database at any time including a full history of past contracts.

Each secretary, for their assigned contracts, will build a schedule for the installers assigned to that same contract. The schedule includes day, time, location, and work required at that location. The installers will have access to their respective schedules so that they can see where they need to go on a given day. After an installer completes a task on their schedule, they can mark that task as completed so that the company and the secretary can track their progress.

The installers require a set of materials to start the installation process. The system maintains a database that keeps track of inventory. If there are sufficient materials in the inventory to conduct the installation, then the inventory manager deducts the used materials from the inventory. If there are insufficient materials, then the inventory manager will order additional materials (outside of the system) and when the materials are delivered, the inventory manager will add the quantity of delivered materials to the database.

Once the installation is complete, after the secretary has verified that the customer is satisfied, the contract is marked as completed.

The CEO oversees adding and removing employees from the system. If a new installer or secretary is hired by the company, the CEO registers them with the system so that they can login with an account and have access to the necessary permissions inside the system.

* + 1. Class diagram

The following diagram describes the types of objects the system interacts with:

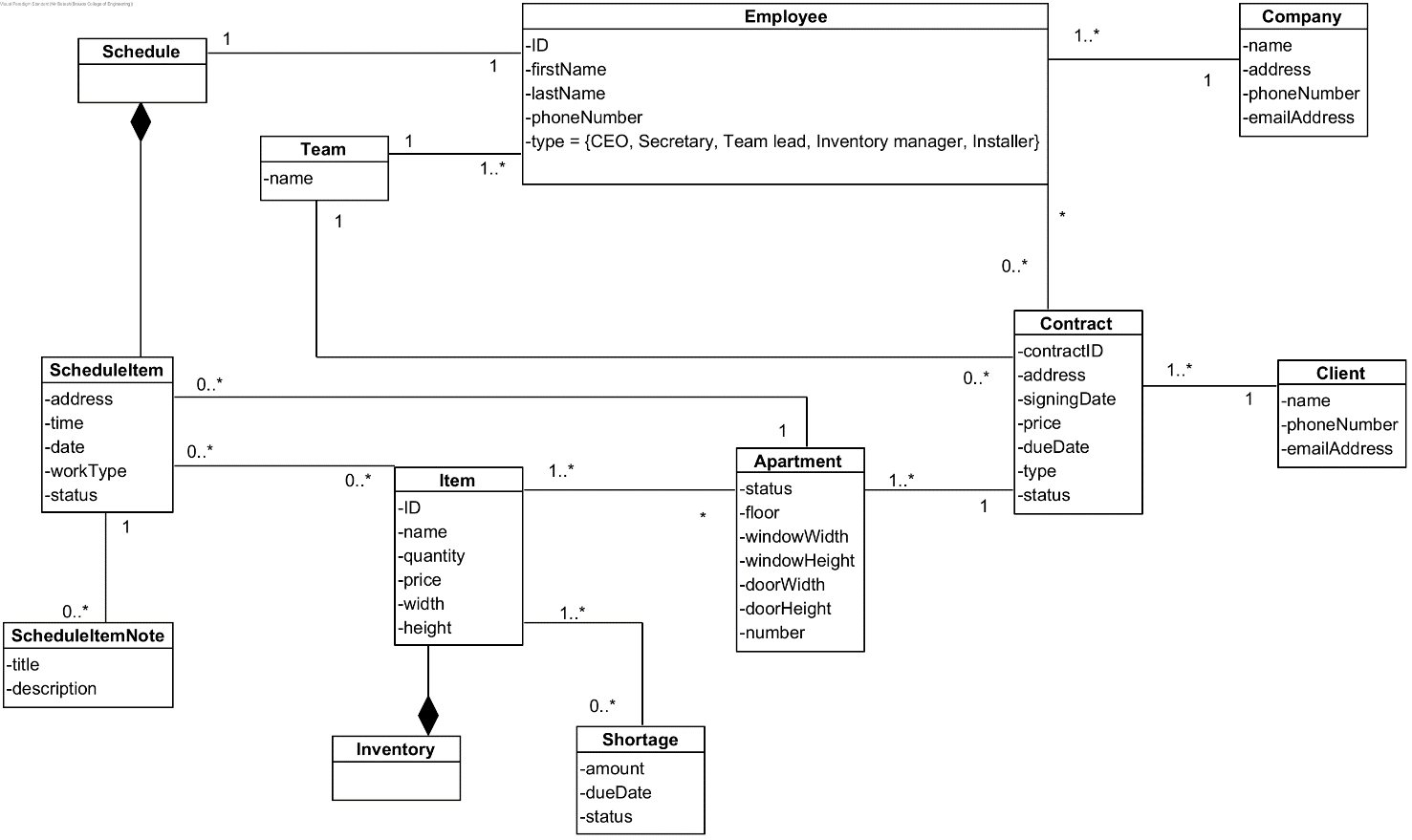


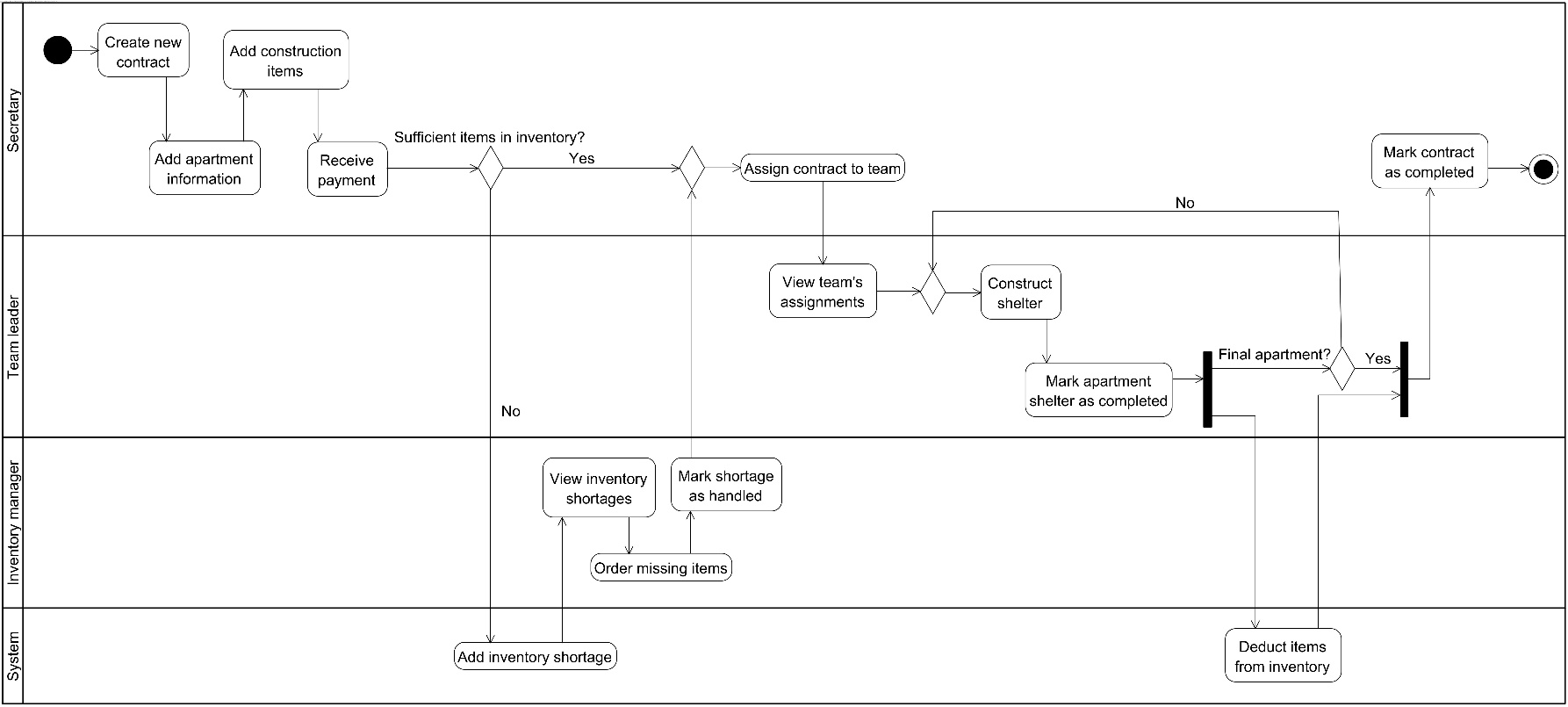
Figure 2: Class diagram

The system is an information system for a shelter construction company that is designed to help manage the company's activities. The company has a name, an address, a phone number, and email address. The company employs the CEO, secretaries, installers, and inventory managers. The employees are identified by their name, ID, and phone number. Each secretary and installer are assigned to a set of construction projects that they manage.

The company signs construction contracts with customers for the construction of shelters. These contracts include location, price, due date, customer's contact information, signing date, contract number and a list of materials needed for the contract.

* + 1. Activity flow chart

The following diagram describes the flow of create contract process:



* + - 1. **Create New Contract** – The process starts with the creation of a new contract.
      2. **Add Apartment Information** – Details about the apartments involved in the contract are added.
      3. **Add Construction Items** – Necessary construction items are listed in the contract.
      4. **Receive Payment** – Payment is received, which triggers the next steps.
      5. Check Inventory:
* **No** (Sufficient Items in Inventory?) – If there are not enough items, the process diverts to manage inventory shortages.
  + - * 1. Add Inventory Shortage: Document or update the system with the shortage information, and the process moves forward to 'assigning the contract to a team'.
* **Yes** (Sufficient Items in Inventory?) – If there are enough items in the inventory, the process moves forward to 'assigning the contract to a team'.
  + - 1. **Assign Contract to Team** – Assign the contract to the appropriate team for execution.
      2. **View Team’s Assignments** – Check which team members are assigned and their specific tasks.
      3. **Construct Shelter** – The team works on constructing the shelter.
      4. **Mark Apartment Shelter Completed** – Each apartment's completion is confirmed.
      5. Check **Final Apartment**:
* **Yes** **(Final Apartment?)** – If it's the final apartment, proceed to mark the contract as completed.
* **No** **(Final Apartment?)** – Deduct the used items from the inventory and continue with the next apartment.
  + - 1. **Mark Contract as Completed** – Once all apartments are completed and the final checks are done, the contract is marked as completed.
    1. Requirements
       1. Functional

|  |  |
| --- | --- |
| **1** | The system allows the CEO to add and remove employees, including secretaries and installers. |
| **2** | The system enables employees to log in with unique credentials. |
| **3** | The system enables secretaries to create, edit, and view construction contracts. |
| **4** | The system stores details for each contract, including location, number of apartments, price, customer contact information, materials needed, and status. |
| **5** | The system allows updating of contract statuses from pending to active upon customer acceptance. |
| **6** | The system tracks and updates inventory levels of materials. |
| **7** | The system generates automatic alerts when inventory levels are low, prompting the ordering of additional materials. |
| **8** | The system allows secretaries to build schedules for installers based on contract requirements. |
| **9** | The system enables installers to access their schedules and mark tasks as completed. |
| **10** | The system updates the progress status of each apartment within a contract based on installer updates and secretary inputs. |
| **11** | The system generates progress reports based on updates from secretaries and installers. |
| **12** | The system provides overviews of active and historical contracts accessible by the CEO and secretaries. |

* + - 1. Non-functional

|  |  |
| --- | --- |
| **1** | Provides a user-friendly interface that is intuitive and easy to navigate for all user roles including CEOs, secretaries, installers, and inventory managers. |
| **2** | Handles multiple user access simultaneously without degradation in performance, ensuring efficient data processing and responsive interactions. |
| **3** | Implements secure login and authentication mechanisms to protect sensitive company and employee information. |
| **4** | Enforces role-based access control, ensuring that employees can only access features and data relevant to their specific roles. |
| **5** | Scales to support an increasing number of users, contracts, and data volume without loss of performance or functionality. |
| **6** | Designed for high availability with minimal downtime to support continuous business operations. |
| **7** | Includes robust error handling and data recovery mechanisms to ensure data integrity and consistent performance. |
| **8** | Easily maintained, allowing for straightforward updates and enhancements. |
| **9** | Supports modular design principles to facilitate future enhancements and integrations without disrupting existing functionality. |

* + 1. Software architecture overview

Our architecture consists of several components:

* A web app through which all users of the system will interact with it. This is the main interface of the system;
* A server hosted in a cloud service. The server handles all data management

Web app

Login

Contract management

Inventory management

Schedule management

Web server

HTTP

User verification

Database management

Figure 3: Architecture overview

* + - 1. Web app

The web app is the main interface of the system, all users’ interactions with the system will go through the web app. The goal is to streamline the use of the system so that it’s as easy as possible to train employees to use the system, even those that are not particularly tech savvy.

The interface will be built using standard web technologies such as HTML5, CSS3 and JavaScript/TypeScript. We will use modern web development frameworks such as SvelteJS which are built on the principle of reactive components. These components will be small reusable pieces of UI that are mostly self-contained that allow fast iteration. Additionally, we will use tailwindCSS to style our UI which will enable simple fast iteration on the styling of the UI where the styling is co-located with the content and logic of each self-contained component.

The app will start out with a login screen to identify the employee. Since each employee has different permissions and responsibilities, an account system will ensure that each employee sees only the things that concern them and not anything else.

Once the employee is logged in, they will be presented with a simple screen showing relevant actions that that employee can take as described in the Use Case diagram. For example, the inventory manager can view the current inventory, a list of item shortages that need to be addressed and an option to mark them as handled.

The UI will be structured in such a way that it is impossible for an employee to take an action that doesn’t make sense or take an action out of order. For example, it will be impossible for a secretary to mark a contract as complete until the contract is marked as having received payment.

* + - 1. Web server

The web server will manage all data in the database and synchronize employees’ access to the system to insure there are no conflicts. The server will be built using a full-stack framework such as SvelteKit and the communication between the client and the server will be done using HTTPS via SvelteKit’s data communication features.

The server will be simple, mostly just fetching data from the database and sending it to the client. The server's purpose is to centralize handling data and ensure that the system is accessible from multiple locations including by team leaders out in the field doing construction work.

The database will contain data similar to the information presented in the Class diagram. The server will ensure that the database is always in a consistent state and will perform regular backups so that in the event of a failure, the system will be able to restore the data to a known good state.

1. **Evaluation / Verification Plan**
   1. Approach to testing
      1. Iterative Testing

The development is broken down into manageable iterations or sprints, each resulting in a potentially shippable product increment. Testing is conducted at the end of each iteration to validate the work and refine or re-prioritize the backlog.

**Application with Jest:** After each sprint, Jest will be used to run all unit and integration tests to ensure new changes have not broken existing functionality (regression testing) and that new features behave as expected.

* + 1. Test-Driven Development (TDD)

In TDD, test cases are developed before the actual software components. These tests define desired improvements or new functions and initially fail. As the development progresses, code is refined until the tests pass. This ensures that the software is built to meet precisely defined requirements from the outset.

**Application with Jest:** Using Jest, tests for each functional component of the web application will be written prior to the implementation of these components. This includes business logic, data validation, and UI component behavior.

* + 1. Unit Testing

We will use Jest for all unit tests.

Scope: Each module, component, and utility function will have corresponding unit tests to check individual functionalities in isolation.

Example: Testing a function that calculates the cost of materials for a project, ensuring that it returns the correct total when given a list of items and quantities.

* + 1. End-to-End Testing

Tools: Although traditionally outside the scope of Jest, basic end-to-end testing can be simulated by testing sequences of actions that span multiple modules.

Scope: Ensure that the complete flow from the front end to the back end works as expected for critical processes.

Example: Simulating the complete process of a user signing a contract, from filling out forms to the submission and confirmation processes.

* + 1. Performance and Stress Testing

Test the system under high load conditions to ensure it can handle multiple simultaneous operations across different modules. E.g. many teams tried to mark as complete when they have finished the construct.

* + 1. Regression Testing

Continuously re-test the system functionalities after every change in the codebase to ensure new changes do not break existing functionalities.

* 1. Detailed Test Plan
     1. Unit Test

|  |  |  |
| --- | --- | --- |
| Component | Test Case | Expected Result |
| User Login | User attempts to log in with valid credentials. | User is successfully logged in and redirected to the appropriate dashboard. |
| Employee Management | CEO adds, removes, or modifies an employee. | System updates employee records and reflects changes immediately. |
| Contract Management | Secretary creates, updates, or completes a contract. | Contract status is updated in the system and visible to authorized roles. |
| Inventory Management | Inventory manager updates or checks inventory levels. | Inventory levels are accurately adjusted and displayed in real-time. |

* + 1. Manual Testing

|  |  |  |
| --- | --- | --- |
| Test Case | Steps | Expected Results |
| Employee Role Management by CEO | CEO logs in, navigates to employee management, adds/removes/modifies employee. | CEO views, modifies employee data; system confirms changes. |
| Contract Management by Secretary | Secretary logs in, creates a contract, updates it to active, schedules installers, marks as completed. | Secretary can manage contracts fully with updates reflected in system. |
| Inventory Management | Inventory manager logs in, checks materials, updates inventory, deducts materials. | Inventory accurately tracked and adjusted with system updates. |
| Installer Task Management | Installer logs in, views tasks, marks task as completed. | Installer sees schedule, updates task status correctly; system tracks progress. |
| Contract Viewing and History | CEO/secretary logs in, navigates to contracts, views active/completed contracts, accesses full history. | Contracts are displayed with options to view historical data. |

**Bibliography**

* + - 1. Enterprise Resource Planning by Al-Amin, Tanjim Hossain, Jahidul Islam and Sanjit Kumar Biwas. (2023)