COS 730 - ASSIGNMENT 2

Contract Instruction Application

System Requirements Specification

(Architectural Specification)

KYLE GAUNT - u15330967

Contents

1 Introduction

CONTRACT INSTRUCTION

A written instruction issued by or under the authority of the principal agent to the contractor, which may include drawings and other construction information.

[Com14]

A Contract Instruction (henceforth CI) has an impact on the various resources of a construction project, be it time, materials or monetary. Due to the importance of the CI in the role of construction, it is vital that construction professionals keep track of and adhere to its contents.

Currently, the process of issuing, editing and interacting with CIs is still done with pen and paper on site, typed up and finalized at the creator's place of work and then distributed using email chains between all involved parties. This means that all relevant information is provided in plain text and must therefore be organised into their relevant categories by each person involved. There is no error checking or notification system to ensure that CIs are being created accurately and alerting users as soon as one is created.

There is a significant space for innovation and a bespoke application that can improve the process from the creation on site all the way through to the distribution of the final document.

In order to solve the aforementioned problems, the Contract Instruction Application has been designed to make the issuing, tracking and closing of CIs as simple and effective as possible. It allows users to send, receive and interact with CIs in real-time and on-the-go, be it in a corporate setting or on an active building site. The project must be accessible on both mobile devices and desktop to cater for all users, be they more office-bound or on-site professionals.

Users can issue a new instruction, specify which areas of the project will be affected, and create custom lists of contacts with which the instruction will be shared. Depending on user privileges, the receiver will be able to

interact with the CI accordingly and all relevant parties will be updated in the event of any changes.

The system includes a full reporting suite which will include reports on the project as a whole, as well as user-defined filtered reports on CIs that meet the specified criteria. The reports can be sent out via email to any user(s) specified by the creator of the CI.

2 Non-Functional Requirements

2.1 *QR1* - Availability

- *QR1.1* - The system has to have high availability to handle user operations in real-time due to the urgent nature of the processes and procedures involved.
- *QR1.2* - The system should be available at least 99 percent of the time (approximately 23h45m a day), apart from where network errors are involved.
- *QR1.3* - The system should be able to operate without an active Internet connection as most active construction sites do not always have Internet access freely available. This can be achieved by saving data locally and uploading it when an active Internet connection is detected.

2.2 *QR2* - Performance

- *QR2.1* - The system must perform its operations in a manner that is swift, efficient and accurate to ensure that any changes made are done so as instantaneously as possible, with minimum strain on the system and with no concern regarding authenticity and integrity.
- *QR2.2* - In order to appear instantaneous, system reaction times must ideally be less than 0.1 seconds, with a maximum response time of under 1 second. This does not included network response times.
- *QR2.3* - Network response times must be no longer than 10 seconds, but ideally less than 5 seconds.

2.3 *QR3* - Scalability

- *QR3.1* - The system should be able to scale in order to accommodate additional/growing user operations, especially during peak work hours. Off-peak hour reduction of resources would be an advantageous feature for the system.
- *QR3.2* - The system will be deployed on an automatic scaling infrastructure in order to allow for efficient and easy scaling, as well as state recovery should the application crash. More resources can be allocated to the system dynamically as needed.

- *QR3.3* - When the system has less than 1,000 users online concurrently, the resources allocated to the system should decrease. Similarly, when more than 1,000 but less than 10,000 users are online concurrently, more resources should be mad available for the system to keep up with the demand. This should scale all the way up to as far as The tiers will be decided according to the policies of the technologies described in 3.6.

- *QR3.4* - With all automatic scaling in place, the system would need to be able to handle at least 100,000 users per second on average. This number allows enough resources for the system to operate nominally but limits resources enough to waste them unnecessarily.

2.4 *QR4* - Maintainability

- *QR4.1* - The system structure will be modular to adhere to the concept of low coupling and high cohesion. This would help its maintainability because when updating a specific module within the system, the changes remain localized instead of promulgating changes everywhere throughout the system. Once the changes have been made and tested, they can be seamlessly integrated into the live system.

- *QR4.2* - A coding standards document will be included to ensure that all parties follow the same coding procedures and protocols. This will make the system easier to understand and increase the ability for multiple parties to operate on the project in conjunction with one another.

- *QR4.3* - An update of latest fixes/additions must be made available on the application's source control platform in order for teams working on the project to have access to the content of the project as well as any important updates.

2.5 *QR5* - Security

- *QR5.1* - Secure login will be required for all users. This will be done by means of a third-party security application which will be discussed in 3.6.

- *QR5.2* - A secure login system will be implemented and private user information would have to be secured i.e information that would be used for authentication purposes like email addresses.

- *QR5.3* - Information regarding the projects and the information associated with them will also need to be secured to prevent any potential breach of Intellectual Property or Trade Secret laws being broken.
- *QR5.4* - Login must allow for, on average, at least 100,000 users to login to the system concurrently without compromising any personal data/login credentials.
- *QR5.5* - The login process must be completed within 3 seconds for any user.

3 Architectural Design

In order to fully accomplish the task of determining and defining the architectural design of the CI Application, we must look at the various subsections of the design process.

These subsections (discussed below) are important in providing the best opportunity for the system to be designed correctly and effectively from the start and reduce the risk of having to redesign or retire the software in the near future.

The CI Application's design has been carefully considered to create a system that can thrive in the uniquely challenging environment in which it will be used.

3.1 Architectural Patterns

A number of architectural patterns will be implemented in the design of the CI application.

This will increase the system's cohesion whilst reducing coupling. The use of industry standard and widely adopted patterns also improves the system's maintainability for developers.

The CI application is primarily an interactive system. Therefore, the system will be designed using a **3-tier architectural style** at a high level. This

will be comprised of a Presentation, Application and Data tier. Each tier performs a specific function:

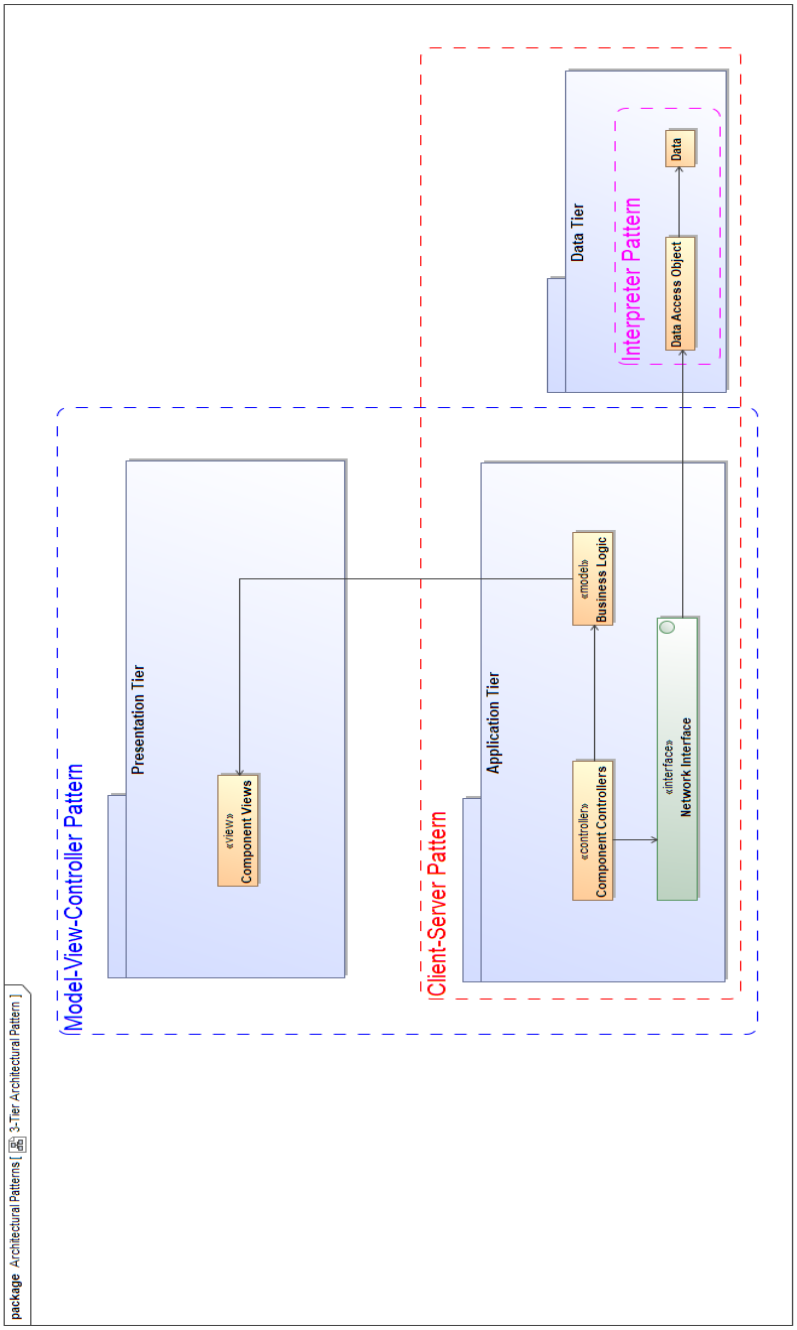
- Presentation tier: everything the user sees/hears/interacts with can be encapsulated in this tier
- Application tier: the functionality behind the Presentation tier is found in this tier
- Data tier: any stored information is retrieved from this tier by the Application tier

This will ensure that the system follows a design principle that allows it to be highly available for the user (as defined in 2.1).

The system will be sub-categorized into the following patterns for the various levels of granularity:

- Presentation tier: Model-View-Controller pattern
  - User interacts with View
  - Controller handles user View input
  - Model performs application's core functionality
- Application tier: Client-Server pattern
  - Used to send and receive information to and from the database
  - Each user needs access to the database from their device
  - User interaction with application handled on device
  - User requests sent to database
  - Database returns requested information
- Data tier: Interpreter pattern
  - Implemented for the purpose of querying the database
  - Rules for evaluating expressions of the database query language

Please see below diagram for more details


package Architectural Patterns [3-Tier Architectural Pattern]
Model-View-Controller Pattern
Presentation Tier
«view»
Component Views
Client-Server Pattern
Application Tier
«model»
Business Logic
«controller»
Component Controllers
«interface»
Network Interface
Data Tier
Interpreter Pattern
Data Access Object
Data

3.2 Architectural Styles

3.2.1 Availability

- The system will store information online so that it is synchronized across all users.
- The system will have offline capabilities that will allow it to have limited functionality while the user is without an active Internet connection. This will use local JSON storage to hold the pending data in a queue and will then upload the file when Internet becomes available.

3.2.2 Performance

- The user will be able to simultaneously upload data in the background and perform tasks in the foreground.
- Server synchronization will only take place when prompted by the user. This will reduce processing strain on the system by eliminating constant polling.

3.2.3 Scalability

- The system will monitor the number of users to determine the resource requirements for the current active user count.
- More resources will be allocated when more users are online and vice versa.

3.2.4 Maintainability

- Documentation will be in the form of: Software Requirements Specification, Coding Standards Document and a User Manual.

3.2.5 Security

- User authentication will be present to ensure that information is correctly being distributed to the relevant users.
- Authorization will set permissions for different user types to allow various levels of access and interaction with the system.

- The system will have a built in logging suite to track all user interactions. This will be behind the interface and so the user experience will not change at all to accommodate this functionality.

3.3 Architectural Constraints

3.3.1 Policy

- The system will be subject to strict coding standards and regulations.
- The system will need to be developed and maintained according to the schedule of the team/individual involved in the project.

3.3.2 Equipment

- The system will be designed in line within the limits of the development team/individual.
- Server design/interaction and other time or financially dependent design considerations will be handled as much as possible by third party companies and cloud services.

3.3.3 People

- The person/people working on this system must be able to communicate and work in unison.
- The system must be developed by developers who are familiar with the technologies and processes with which it was designed.

3.3.4 Market

- The system must be able to cater for the size of the market for which it is intended.
- Whether the intended market sees an increase or a decrease in revenue or popularity, the system must react accordingly so as to function effectively operate within the ever changing boundaries of that market.
- The system must be flexible to adapt to the needs of the market for which it has been built. If users require new functionality or features,

the system must have the capability to be changed to meet those needs.

3.4 Actor-System Interaction Models

3.4.1 Actor-System Interaction Model 1

This Actor-System interaction model focuses on the CI subsystem which is responsible for handling the creation of new Contract Instructions. The actor, in this scenario, is a generic Contractor.

Precondition: this use case assumes that the Contractor User has logged into the system and is shown the Contractor main page.	
Actor: Contractor User	System: CI
	0. System displays application main page
1. TUCBW contractor user clicks the "New Contract Instruction" Button	2. System displays the Create New Contract Instruction Page
3. Contractor enters contract instruction details and clicks the "Submit" button	4. System validates the information submitted by the user and displays a popup confirmation message if no errors are found. If errors are found, throw "Error, please review the document for more details" and return user to the previous page.
5. TUCEW the contractor clicks the "Okay" button	6. System returns the user to the main page
Postcondition: the new contract instruction is immediately searchable for and edited	

3.4.2 Actor-System Interaction Model 2

The actor, in this scenario, is a generic Contractor who is going through the process of editing an existing Contract Instruction.

Precondition: this use case assumes that the Contractor User has logged into the system and is shown the Contractor main page.	
Actor: Contractor User	System: CI
	0. System displays application main page
1. TUCBW contractor user clicks the "View Contract Instructions" Button	2. System displays the All Contract Instructions Page
3. Contractor clicks on contract instruction	4. System displays the Edit Contract Instruction Page
5. User edits contract instruction details	6. System validates the information submitted by the user and displays a popup confirmation message if no errors are found. If errors are found, throw "Error, please review the document for more details" and return user to the previous page.
7. TUCEW the contractor clicks the "Okay" button	8. System returns the user to the main page
Postcondition: the contract instruction is immediately edited and displayed.	

3.5 Deployment Model

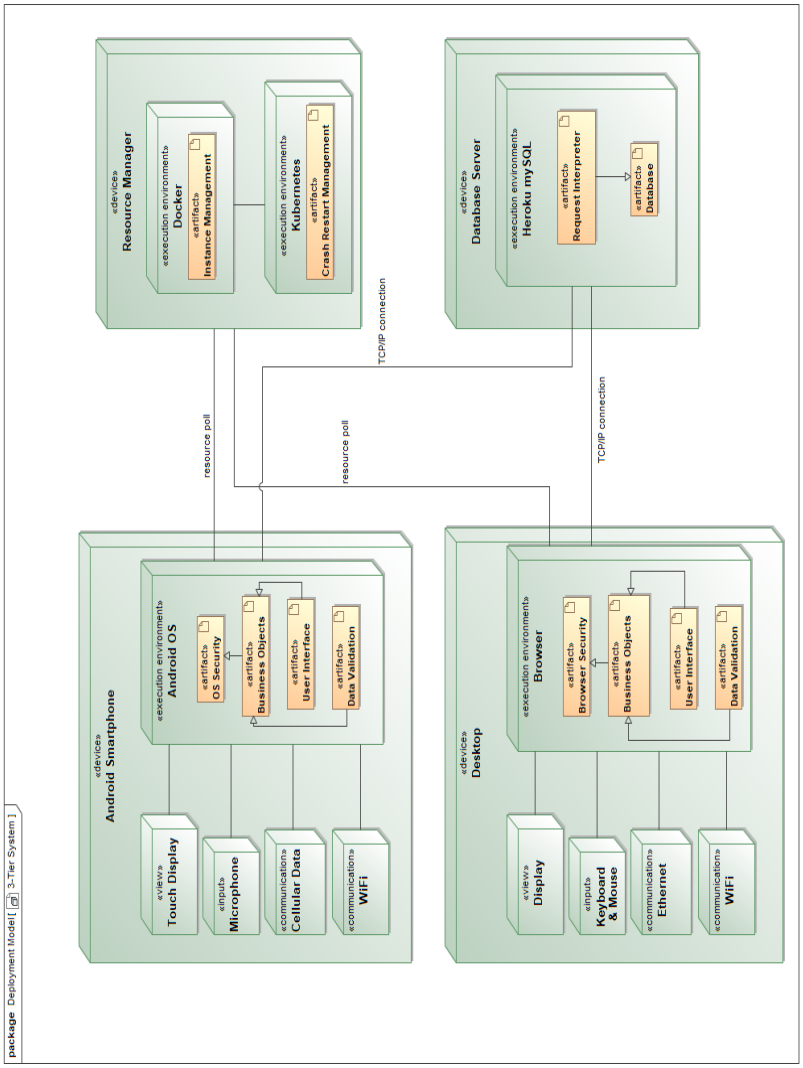
The system will be designed using a 3-tier architecture pattern which means that each subsystem must be capable of being self-sufficient and not reliant on any other subsystem. To achieve this, the system has been designed with low coupling and high cohesion in mind.

Two user interfaces (Presentation tier) are supported (Android and web browser compatible version) which both communicate with the database independently of one another. All database queries are performed using TCP/IP protocol. This protocol is preferable to UDP as the information needs to be secure and accurate for all users on the system.

The database (Data tier) makes use of an Interpreter pattern to make queries to the mySQL database that is supported by Heroku (the cloud based hosting service).

Finally, the Application tier is responsible for communicating with the re-

source manager to request more or less resources according to the current system load.



3.6 Technical Requirements (Technology)

- Presentation Layer

  - The user interface will be handled by an MVC pattern which will be provided in the form of an Android application as well as a web browser compatible version that will be designed and built using HTML and CSS.

- Application Layer

  - The Android version will be built natively in Android Studio using the Java programming language, XML and various Android application libraries which will handle all business logic.

  - The web browser version will run Python as a business logic facilitator.

  - The system will interact with Docker and Kubernetes to control resource management.

- Data Layer

  - This layer will use an online database hosting service called Heroku which uses the mySQL query language.