COMP 3059 – Capstone Project I

Software Requirements Analysis and Design Assignment

This assignment is an overview to gather the software needs with requirements analysis and help to proceed with the design.

The requirements analysis helps to break down functional and non-functional requirements to a basic design view to provide a clear system development process framework. It involves various entities, including business, stakeholders and technology requirements.

The design is the activity following requirements specification and before programming. Software design usually involves problem solving and planning a software solution.

To work on this assignment you could use the references and a sample template given below. The sample template can be customised to suit the nature of your project.

Reference Readings/Example:
http://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf

www.cse.msu.edu/~chengb/RE-491/Papers/SRSExample-webapp.doc

Source for this template:
www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc

1.0 Introduction

The Introduction section provides an overview of the system using software requirements analysis and design for the scope of the system.

1.1 Purpose

This document describes the high-level software requirements for the contractor contact system. It describes what, not how, of the capabilities of the system for the intended audiences.

This document servers as a reference for all stakeholder, including clients, project stakeholders, developer, and instructors to ensure a clear and shared understanding of a system required features and behaviour.

This document outlines the system's functional and non-functional requirements, specified system features, and database design. It also includes assumptions and constraints that will guide the system design and future development activities.

1.2 Scope

- **User registration and login**
  - Support account creation and secure authentication for both client(homeowner) and contactor

- **Project posting for Clients (homeowner)**
  - Allow Clients to create project listings with essential details including title, description, location, and budget range.

- **Project Browsing and Search for Contractors**
  - Eanbles contractors to browse and search available projects based on their interests or services area

- **Bid Management**
  - Contractors can submit bids for projects, as well as view, accept, or reject bid decisions
  - Clients can review submitted bids and choose the most suitable option

- **Clients review management**
  - Clients can provide review of contracts works.
- **Payment management**
  - Allow clients to make payments for accepted bids and enables contractors to receive payments through an integrated payment system.
- **Admin Dashboard for Managing Listings and Users**
  - Provides administrator with tools to view and manage user accounts and projects listings.

Upload and view project documents such as estimate PDFs

2.0 System Overview

This section summarizes the project's purpose and context, outlines key stakeholders and goals, and states the main constraints, assumptions, and dependencies that guide our design. It also defines what is in scope for this release and what is not.

2.1 Project Perspective

Contractor Connect is a new, self-contained web application that matches homeowners with local contractors. Today, most people rely on word of mouth, scattered social posts, phone calls, spreadsheets or email to request quotes and keep track of work. Our app replaces that messy process. Compared with local listing boards and gig platforms, Contractor Connect focuses on small and medium home projects, uses structured bids so prices are easy to compare, keeps reviews simple, and includes basic in-app payments. The main stakeholders are homeowners, contractors, an administrator for moderation, the instructor, teaching assistants for review, and our development team.

In this release, we will deliver the essentials: user accounts, project posting, project browsing and search, bidding, bid review and selection, a simple dashboard, reviews, and basic payment handling.

2.2 System Context

Business goals

- Help homeowners get multiple comparable bids quickly.
- Help contractors discover relevant jobs without cold outreach.
- Keep a simple record of projects, bids, payments, and reviews.

Success metrics

- A homeowner can create and publish a project in ≤ 2 minutes (demo conditions).
- A contractor can submit a bid in ≤ 2 minutes once registered.
- ≥ 90% of page requests respond in ≤ 1 second on the demo environment.
- Each posted project receives ≥ 2 bids during test seeding/demo.

Primary actors & interactions

- **Homeowner:** register or log in; post a project with title, description, location, and budget range; receive and compare bids; select a contractor; submit a review; make a payment.
- **Contractor:** register or log in; browse and search projects; place or withdraw bids; view award decisions; receive payment; respond to reviews.
- **Admin:** view and manage users and listings; perform basic moderation and reporting.

External services

- Email notifications for verification, bid updates, and status changes.

- File storage for project images and documents such as estimates and photos.
- Payment processor in sandbox mode for the demo.
- Hosting for the frontend, backend, and database.

Key in-scope flows

- Client posts a project → Contractor's search/bid → Client reviews/selects → Payment captured → Work completed → Client review recorded.

Explicit out of scope (future work)

- Formal contracts and dispute resolution, complex escrow, identity verification, rich messaging with file attachments, mobile apps, and advanced analytics.

2.3 General Constraints

- **Course and timing:** Team of 5, Sprint 3 deliverables, follow the SRS template.
- **Tech:** Desktop first web app. Likely stack: React UI, Node/Express API, MongoDB Atlas DB. Host on student friendly free tiers such as Vercel, Render, or Heroku.
- **Security and privacy:** Hashed passwords, role-based access for Client, Contractor, and Admin, server-side validation, minimal PII, and basic rate limiting on auth/bids.
- **Accessibility and UX:** Keyboard friendly forms, clear error messages, readable contrast, and consistent components using Bootstrap or Material UI.
- **Operational:** Seed data for the demo, provide simple deploy steps, enforce upload limits such as images ≤ 5 MB, and avoid paid only features.

- **Licensing/compliance:** Use permissive OSS where possible and keep payment in sandbox.

2.4 Assumptions and Dependencies

Assumptions

- Users have internet access and a valid email.
- Projects are small to medium home jobs.
- English UI is acceptable for this course.
- Uploaded images/docs are safe and within size limits.

Dependencies (with fallback)

- **MongoDB Atlas:** free-tier limits or outage → local Mongo for demo.
- **Email/SMTP:** throttling → dev mailbox or log-only mode.
- **File storage:** quota issues → local storage for demo.
- **Payment processor (sandbox):** key/limit issues → mocked payment flow.
- **Hosting:** cold starts/timeouts → pre-warm before demo and provide local run steps.
- **UI/libs (Bootstrap/MUI/Router):** version conflicts → pin versions and test before merge.

3.0 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.1 <Functional Requirement or Feature #1>

Feature #1: Account Login / Creation (Client & Contractor)

Introduction
This feature allows new users to create an account, and for existing users to log into their accounts.

Inputs
- email
- password
- userType (*client* or *contractor*)
- username

Processing
- validateInput:
  - Checks that all required fields are completed and formatted correctly.
- verifyLogin:
  - Checks that the input matches an account on file.
- create account
  - Saves user information to the user database.
  - Forwards user to login screen on completion.

Flow:
validateInput > createAccount
validateInput > verifyLogin

Outputs
- Confirmation message: "Logged in successfully ☑".
- New user account added to user database.
- Error if validation fails "Error: Login failed ✗".

Feature #2: Task Creation (User/Client)

Introduction
This feature allows users/clients to create a task and post it for bids.

Inputs
- title
- taskDescription
- taskCategory (*cleaning*, *electrical*, *construction*, etc...)
- budgetRange
- location
- requiredSkills
- desiredStartDate

- deadline
- userId

Processing
- validateTaskInput
  - Checks that all required fields are completed properly.
- saveTask
  - Saves the task information to the database.
- postTask
  - Makes the task visible to contractors on the platform.

Flow:
validateTaskInput > saveTask > postTask

Outputs
- Confirmation message: "Task created successfully ☑".
- Task added to database
- Error if validation fails

Feature #3: Contractor Bidding (User/Contractor)

Introduction
This feature allows contractor type users to bid on a task.

Inputs
- Task ID
- Bid amount
- Estimated start date
- Estimated end date
- Relevant skills / certifications
- Contractor ID
- Contractor Email

Processing
- validateBid
  - Checks that all required fields are completed properly.
- verifyContractorEligibility
  - Checks if the contractor's account is verified and in good standing to bid on a task.
- saveBid
  - Saves the bid information to the database.
- sendBid
  - Sends a response to the client with the details of the bid.

Flow:

validateBid > verifyContractorEligibility > saveBid > sendBid

- Outputs
  - Confirmation message: "Bid submitted successfully ☑"
  - Bid added to database with the related task
  - Bid information sent to client for review.
  - Error if validation fails "Error: Bid could not be placed ❌"

Feature #4: Bid Review / Selection (User/Client)

Introduction
This feature allows client type users to review the bids on their task and select a contractor to award the task.
Inputs

- clientId
- taskId
- contractorId

Processing

- getAllBids
  - Retrieves all bids placed on the task.
- selectBid
  - Sets the selected contractor as active and closes the task from receiving more bids.
- updateTaskStatus
  - Sets the task status to "*Assigned*".

Flow:

getAllBids > selectBid > updateTaskStatus

Outputs

- Confirmation message: "Contractor selected successfully ☑"
- Task status updated to *Assigned*.
- Confirmation message sent to contractor.
- Error if validation fails "Error: Contractor selection failed ❌"

Feature #5: Dashboard (Client & Contractor)

Introduction
This feature provides users with a personalized interface to view and manage their account.

Inputs

- userId
- userType (*client* or *contractor*)

Processing

- loadUserData
  - Retrieves all task and bid information for the selected account.
- loadDashboard
  - Generates the list of tasks and notification icons.
- updateDashboard
  - Updates the dashboard when a new notification is received (Task, bid, or message)

Flow:

loadUserData > loadDashboard > updateDashboard

Outputs

- A personalized user interface.
- Icons and colour accents used to highlight updated messages or notifications
- Error if validation fails "Error: Dashboard failed to load ❌"

3.2 Use Cases

3.2.1 Function 1: Account Login / Creation

1. Shirley Edwards
    - Shirley is a senior homeowner whose go-to handy-man has retired, and now she needs help finding a replacement. Shirley creates an account to use the search feature to find out about contractors in her area.
2. Jordan Carr
    - Jordan is a local contractor looking to make extra cash. Jordan creates a contractor profile to browse tasks that fit his schedule so they don't interfere with his full time job.
3. Marie LaValle
    - Marie is a homeowner with a busy schedule and needs to get the yard work done before the neighbours complain again. Marie logs into her account to see if anyone in the neighbourhood can help.

3.2.2 Function 2: Task Creation

1. Nazrin Saleha
    - Nazrin has uneven stepping stones in the path leading to their home. He doesn't want his kids or visitors to trip and get hurt, so he posts a task to have the stones releveled.
2. Ali Hassan
    - Ali is moving into the area and doesn't know anyone yet. The movers have left everything on his lawn and have disappeared. Ali creates a task to find help getting the job done, which not only gets him moved in, but helps him make his first connections in the area.
3. Margaret Davies
    - Margaret is having her garden features in a home and garden magazine and needs her deck and railings painted before the photoshoot at the end of the month. Margaret creates the task and sets the deadline to ensure the project can be completed on time.

3.2.3 Function 3: Contractor Bidding

1. Joe Leonard
    - Joe is a contractor who sees a task for a garage repair that he is qualified to perform. Joe places a bid on the project with the estimated cost, start and end date.

2. Michelle Yu
    - Michelle is a certified electrician. When she bids on a project to rewire a lightswitch, her skills /certification can be viewed by the client, which gives her a better chance of being selected.
3. Alex Ivanov
    - Alex is bidding on a project that he can't complete before the listed deadline. Alex includes their own estimated end date for the client to review in the bid. After review, the client approves the alternate date and Alex gets the bid!

3.2.4 Function 4: Bid Review / Selection

1. Farzana Solomon
    - Farzana created a task to help clean out her garage and has received three different bids. She reviews the bids and selects one second bid which she finds to be priced fairly and includes a realistic timeline.
2. Steph Wilson
    - Steph has received a bid on her task, but she doesn't want to move forward with the contractor. Without selecting a winning bid, Steph can continue to receive new bids on her task.
3. Matthew McCormick
    - Matthew has received a bid on his task that he is considering. Matthew connects with the bidder to discuss the details further before they come to a final agreement.

3.2.5 Function 5: Dashboard

1. Terek Yacin
    - Terek has two different tasks ongoing with different contractors. When he gets home at the end of the day he wants to quickly check the tasks' status. He visits the user dashboard and can see all the updates and notifications for each of his tasks in one place.
2. Alisha Martin
    - Alisha has multiple tasks ongoing at any given time. She visits the dashboard daily to track and update the progress on their tasks.
3. **Jason Liu**
    - Jason has been waiting for someone to bid on his task. He visits his dashboard and sees that a small red icon indicates he has received a new bid.

4.0 Non-Functional Requirements

The following non-functional requirements will define how the Contractor Connect system must operate, ensuring performance, reliability, security, usability, maintainability, and portability. All requirements are stated in measurable terms.

Legal & Compliance:

- Must use free or permissive open-source tools suitable for academic use.
- Payment processing must remain in sandbox mode with no real financial transactions.
- Only minimal personal information should be collected to meet privacy expectations.
- Uploaded content must comply with acceptable-use standards (no harmful or illegal files).
- System data must be retained for **3 years** for audit and reference purposes

Financial & Operational Constraints:

- System must run entirely on a free hosting site.

- File uploads must stay within storage limits (e.g., **≤ 10 MB per file**).
- Payment features must include mock features when sandbox limits are reached. This will be only for demonstration purposes, and no financial system will be connected, which could hinder our operations.
- Local run instructions must be provided to ensure reliability during demos.

User Login & Account Reliability Constraints

- 99% of valid login attempts must succeed under normal demo conditions.
- Dashboard and primary pages must load in ≤ 1 second for typical data volumes.
- Role-based dashboards (Client, Contractor, Admin) must load accurate data consistently.

Security & Privacy Constraints

- All passwords must be securely hashed before storage.
- Role-based access control must prevent unauthorized access to Projects, Bids, and Admin tools.
- All sensitive operations must use HTTPS, secure sessions, and server-side validation.
- Input validation must prevent SQL/NoSQL injection and cross-site scripting (XSS) issues.
- File uploads must be scanned/validated to avoid malicious content.
- Database operations must follow ACID properties to maintain data integrity.

5.0 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc?

- Database usage
  - A relational database will be used to store system data such as user information, project information, and other data.
- Data format
  - Data will be stored in structured tables using standard SQLdata types.
  - Fields will be stored using a consistent format to make sure the data will be accurate and readable.
  - Data encoding will follow UTF-8 to support multiple language.
- Storage Capabilities
  - Database must support scalable storage for future growth
  - Auto backup procedures will be used to eleminate the risk of losing data
- Data Retention
  - System data will be stored in an archive for 7 years in case they are needed.
- Data Integrity
  - To ensure relation integrity a primary and foreign key will be used
  - Validation rules will prevent duplicate or invalid entries.
  - Transaction will follow ACID properties.

- Security and Access Control.
    - Encrypted passwords will be used for accessing data
    - Authentication and authorization will be implemented at the database level.

6.0 Other Requirements

Additional requirements, if any.

7.0 Approval

The signatures below indicate their approval of the contents of this document.

Project Role	Name	Signature	Date
Project Manager	Adel Alhajhussein	*Adel Alhajhussein*	2025-11-15
Front End Developer	Sana Karnelia	***sana karnelia***	2025-11-16
UI Design	Shifa Karnelia	Shifa Karnelia	2025-11-16
Quality Assurance	Eric Laudrum	Eric Laudrum	2026-01-24
Database	Mehdi jazi	Mehdi jazi	2026-01-23