Ainjazha management system

Software Requirements Specification

VERSION:1.0

2024

Taha Shabaan
Lead Software Engineer

Table of Contents

1. Introduction

The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document, remember that it should contain all the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in it.

1.1 Purpose

This system helps connect between Artificial and customers

1.2 Scope

This subsection should:

- **User** registration and authentication.
- **Task** creation and management.
- **Tasker** search and bidding.
- **Secure** payment processing.
- **Ratings** and reviews.
- **Customer** support and dispute resolution.1.3 Definitions, Acronyms, and Abbreviations

Definitions	Acronyms
kadam	task
administration	admin
manger	manger
Artificial	Taskers
customers	uaers

2. General Description

This section of the SRS should describe the general factors that affect 'the product and its requirements'. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.

2.1 Product Perspective

The Ainjazha task Management System is a web and mobile-based platform designed to facilitate the efficient and convenient outsourcing of tasks and services within a local community. Ainjazha connects people who need assistance with various tasks and errands with skilled individuals willing to offer their services, creating a marketplace for on-demand assistance.

2.2 Product Functions

The system includes some feature such Task Posting and Browsing, user profile, matching and assignment task, mange task and tasker, real-time-message, rating, and review

2.3 User Characteristics

user	role
customers	These users are individuals or businesses looking for help with various tasks or services and you can add review and rate
Taskers	These users are individuals who offer their skills and services to complete tasks and you can add review and rate
Support Seekers	Some users may require assistance with the platform
admin	The admin is responsible for managing and maintaining the khadam platform

2.4 General Constraints

- Compliance with local, national, and international laws and regulations, including labor laws, tax regulations, and data privacy laws.
- Licensing and insurance requirements for certain services or tasks, especially those related to health, safety, or professional services.
- Ensuring the security of user data and financial transactions, including protection against data breaches and fraud.
- Compatibility with various operating systems, web browsers, and mobile devices to ensure a broad user base.
- Technical constraints related to the scalability and performance of the platform, especially during peak usage times.

3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be:

- Correct
- Traceable (both forward and backward to prior/future artifacts)
- Unambiguous
- Verifiable (i.e., testable)
- Prioritized (with respect to importance and/or stability)
- Complete

- Consistent
- Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefully organized requirements presented in this section so that they may easily be accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and hence design) the software project within this SRS.

3.1 External Interface Requirements

3.1.1 User Interfaces

1. **Web Application Interface**
    - The web application UI allows users to access khadam from their desktop or laptop web browsers.
2. **Mobile App Interfaces**
    - Khadam offers mobile applications for both iOS and Android platforms, each with its own user interface tailored to the respective operating system's design guidelines. Mobile app interfaces include task discovery, task posting, communication with taskers, location-based services, and task management.

3.1.2 Software Interfaces

1. Payment Processing Services:
    - Payment gateways and processors like PayPal, Stripe, or Square for handling financial transactions between users and taskers.
2. Mapping and Location Services:
    - Mapping and location-based services such as Google Maps or Mapbox to display task locations, calculate distances, and provide navigation directions
3. Calendar and Scheduling Tools:
    - Calendar integration services like Google Calendar or Apple Calendar to assist with scheduling tasks and managing appointments.
4. User Authentication and Authorization:
    - Identity verification services for user authentication, ensuring that users are who they claim to be. This might involve services like OAuth or identity verification provider
5. Content Delivery Networks (CDNs):
    - CDNs like Amazon CloudFront or Akamai for delivering images, media, and other content efficiently to users across the globe

3.1.4 Communications Interfaces

1. In-App Messaging
   - Khadam includes an in-app messaging system that allows users to communicate directly within the platform. This feature facilitates real-time text-based conversations between task posters and taskers
2. Notifications and Alerts
   - Khadam can send notifications and alerts to users' devices to keep them informed about task updates, new messages, or other relevant activities.
3. Voice and Call Features
   - Khadam might provide voice communication options, allowing users to make voice calls or engage in voice chats if necessary for task-related discussions

3.2 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.

3.2.1 User Registration and Authentication:

●User Registration

- **Form**: Design a user registration form capturing essential details (e.g., name, email, password). Consider adding optional fields (e.g., phone number, profile picture).
- **Social Media Login**: Implement options for social media login (e.g., Google, Facebook) to enhance user convenience.
- **Validation**: Ensure validation checks for accuracy and completeness of user-provided data.
- **Security**: Implement password encryption (e.g., hashing) for secure storage.
- **Email Confirmation**: Include an email confirmation process to verify the user's email and activate their account.
- Email Verification: Send a confirmation email for email verification and account activation.
-

User Authentication

- **Secure Login**: Allow users to log in using their email and password, with support for social media logins.
- **Session Management**: Implement session handling, including automatic logout after inactivity.
- **Password Recovery:** Provide mechanisms for password recovery and account unlocking.
- **Two-Factor Authentication** (2FA): Consider adding 2FA for enhanced security

3.2.3 User Profiles

- **Privacy Controls**: Implement privacy settings for controlling profile visibility.
- **Profile Management:** Enable users to update their account information and delete their accounts if needed.
- **Communication Preferences**: Provide options for managing notification preferences.
- **Account Management:** Implement features for updating account information, deleting accounts, and managing communication preferences.

3.2.3 Tasker Profile and Skills:

- **Skill Showcase**: Enable taskers to highlight their skills, certifications, and previous work through a portfolio feature.
- **Profile Creation**: Allow users to create and personalize profiles with additional information (e.g., bio, skills, location).
- **Pricing and Bidding**: Allow taskers to set their pricing or bid on tasks, facilitating negotiation between task posters and taskers.
- **Portfolio Feature**: Enable taskers to upload portfolios of previous work, accessible to task posters for skill assessment.

3.2.4 Task Posting (optional):

- **Task Form:** Allow users to post tasks with details like title, description, location, date, and budget.
- **Validation & Formatting**: Implement validation checks and provide a text editor for detailed task descriptions.
- **Location Input:** Include location fields with address autocomplete and optional map integration.
- **Task Scheduling**: Offer date/time pickers and options for flexible or recurring schedules.
- **Attachments:** Allow file uploads with size/type limitations, and preview tasks before submission.

- **Task Management**: Save tasks, provide an interface for task tracking, and allow task edits or deletions.
- · **Task Status System:** Implement statuses like "Pending," "Assigned," "In Progress," "Completed," and "Cancelled."
- .

3.2.5 Task Categories

- **Task Categories**: Offer predefined task categories and subcategories for better task classification.
- **Skills & Qualifications**: Include options for specifying required skills or qualifications.

3.2.6 Searching and Browsing:

- **Search Functionality**: Implement a search bar with keyword matching and advanced search options (e.g., filtering by location, date, budget).
- **Result Display**: Use pagination or infinite scrolling for search results, with sorting options (e.g., relevance, date, budget).
- **Advanced Search**: Offer filtering by location, date, or budget, and include pagination or infinite scrolling.
- **Sorting Options**: Provide options to sort search results by relevance, date, budget, or tasker rating.

3.2.7 Matching and Assignment Task

- **Profile and Task Details**: Ensure taskers can create profiles showcasing their skills, and task posters provide detailed task descriptions.
- **Matching Algorithm**: Develop an algorithm considering factors like task category, skills, location, and ratings to match taskers with tasks.
- **Communication**: Provide in-platform messaging for task posters and taskers to discuss tasks and finalize details.

- P**ost-Task Review:** Allow task posters to review and rate taskers post-completion.
- **Assignment & Status Update:** Confirm assignments and update task statuses upon assignment.
- **Recommendations**: Provide tasker recommendations to task posters, allowing profile exploration.
- .

3.2.8 Tasker Ratings and Reviews

- **Rating System**: Implement a rating and review system for completed tasks, allowing task posters to rate and comment on taskers.
- **Review Display**: Display average ratings and individual reviews on tasker profiles, with options for taskers to respond.
- **Review Process:** Prompt task posters to leave feedback, with a defined rating scale (e.g., 1-5 stars).
- **Tasker Responses**: Allow taskers to respond to reviews on their profiles.

3.2.9 In-App Messaging and Communication

λ **Messaging Feature**: Implement encrypted messaging for secure communication between taskers and task posters.

λ **Real-Time Messaging**: Utilize technologies like web sockets or push notifications for instant messaging.

λ **Multimedia Support**: Allow exchange of text messages and multimedia content (images, documents, videos) within the messaging interface.

λ **Search & Navigation**: Provide search functionality and filters within the messaging system.

λ **User Information**: Display basic user information within the messaging interface for easy reference.

3.2.10 Notifications and Reminders

- **Customizable Notifications**: Allow users to customize notification settings and send reminders for task updates, new messages, or system maintenance.
- **Task Updates**: Notify users about task updates, including status changes and new task postings.

- **Message Notifications**: Send notifications for new messages and conversation updates.
- **Feedback Requests**: Prompt users for ratings and reviews post-task completion.
- **Maintenance Notifications**: Inform users about scheduled maintenance, system updates, or downtime.
-

3.2.12 Mapping Services:

- **Location Input**: Provide options for entering task locations, including map pins and saved locations.
- **Geo-Fencing**: Implement geo-fencing to filter tasks visible to nearby taskers.
- **Real-Time Updates**: Use real-time tracking to update tasker locations on the map.
- **Map View**: Offer a map view of tasks, with distance filters for taskers.

3.3 Use Cases

Use case name	User Registration
Actors	Client, tasker
Description	Tasker and customer, you can authentications
Trigger	
Preconditions	No
Normal Flow	Take data from customer and tasker Check data it valid or not
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	User Authentication
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	

Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	User Profiles
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	Task Posting
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	Searching and Browsing
Actors	
Description	
Trigger	

Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	Matching and Assignment Task
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	Tasker Ratings and Reviews
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	In-App Messaging and Communication
Actors	

Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

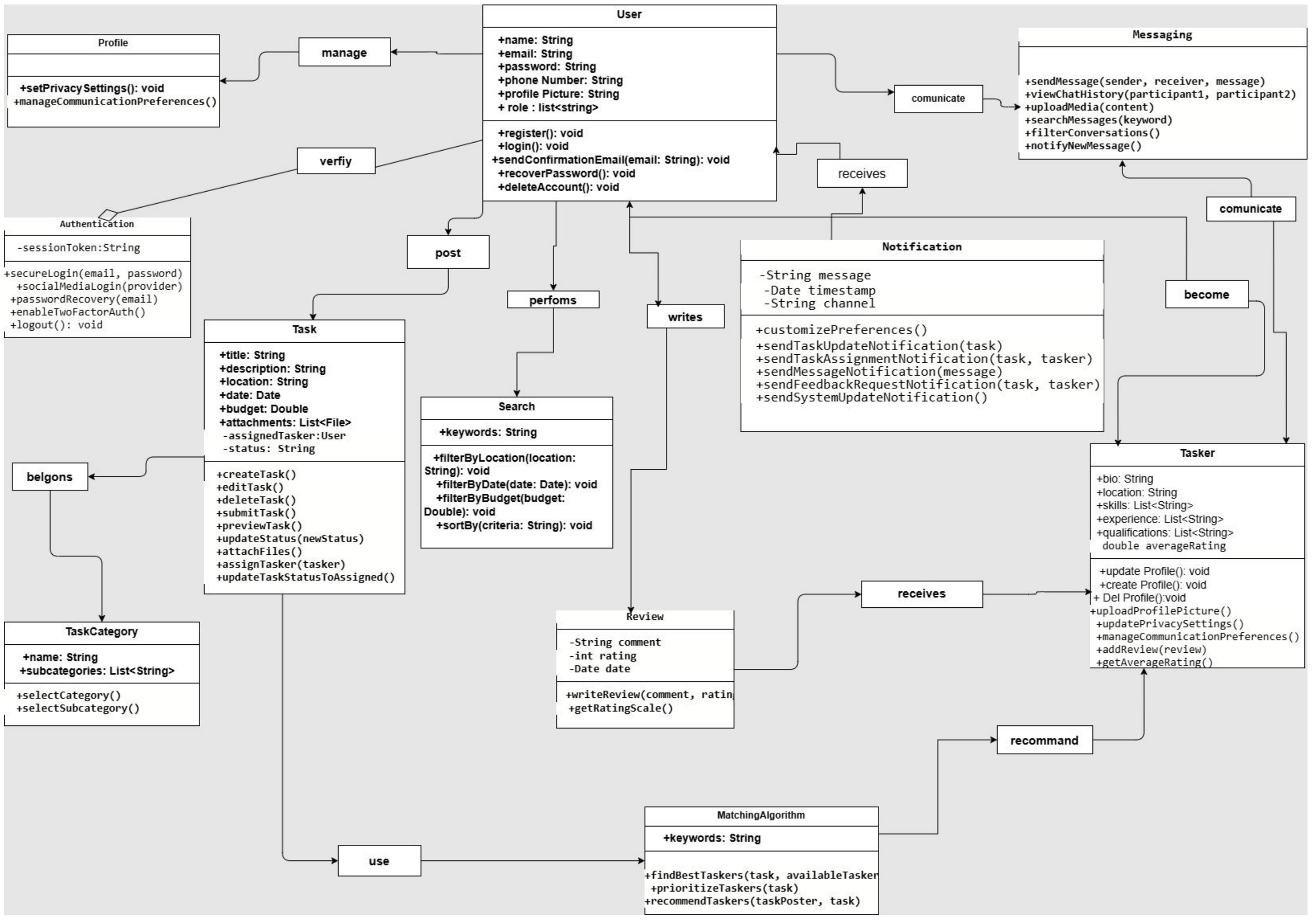
Use case name	
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	Notifications and Reminders
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	Tasker Profile and Skills
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

Use case name	**Mapping Services**
Actors	
Description	
Trigger	
Preconditions	
Normal Flow	
Post conditions	
Exceptions	
Alternative Flows	
Summary Inputs	

3.4 Classes / Objects



3.4.1 <Class / Object #1>

3.4.1.1 Attributes

3.4.1.2 Functions

<Reference to functional requirements and/or use cases>

3.4.2 <Class / Object #2>

…

3.5 Non-Functional Requirements

3.5.1 Performance

- **Load Handling**: System must support 10,00 concurrent users with a response time of less than 2 seconds.
- **Caching: Implement** server-side and client-side caching to improve performance.

3.5.2 Reliability

- **Accessibility Compliance**: Ensure the platform meets WCAG 2.1 standards.

- **Intuitive Interface**: Design for ease of use with minimal clicks required to complete core actions.

3.5.3 Availability

- : **Redundancy** Implement failover systems and data backups to ensure 99.9% uptime.

3.5.4 Security

- **Data Encryption**: Use SSL/TLS for data transmission and AES-256 for data storage.
- **Regular Audits**: Schedule regular security audits and penetration testing.

3.5.5 Maintainability

- **Separation of Concern**s: Ensure the platform's codebase is modular, with clear separation of concerns (e.g., UI, business logic, data access). This allows different teams to work on separate modules without causing conflicts.

:

- **Inline Documentation**: Use clear and consistent inline comments to explain complex sections of the code. This will help developers understand and maintain the codebase over time
- **Unit Tests**: Implement unit tests for individual components to ensure that changes do not introduce bugs. Aim for high test coverage across the codebase.
- **Integration Tests**: Develop integration tests to verify that different parts of the system work together as expected.

3.5.6 Portability

- **Standardized API Design**: Design APIs according to industry standards (e.g., REST) to ensure that they can be consumed by different clients or integrated with various third-party services without requiring changes.
- **API Versioning**: Implement versioning for APIs to ensure backward compatibility, allowing clients to continue functioning even if the API is updated.

- **Database Independence**: Design the system to work with different types of databases (SQL, NoSQL) with minimal changes. Use an abstraction layer for database operations to facilitate this portability.
-
- **Data Migration Tools**: Implement tools and scripts for migrating data between different database systems, ensuring that the platform can easily switch from one database technology to another if needed.

- **Frontend Compatibility**: Ensure that the frontend is compatible with multiple browsers (Chrome, Firefox, Safari, Edge) and mobile operating systems (iOS, Android). Use responsive design principles and testing tools like BrowserStack.

- **Multiple Environments**: Ensure the platform can be easily deployed in different environments (development, staging, production) with minimal configuration changes. This includes supporting both on-premises and cloud-based deployments.

3.6 Inverse Requirements

State any *useful* inverse requirements.

3.7 Design Constraints

Specify design constraints imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

3.8 Logical Database Requirements

Under test (not complete)

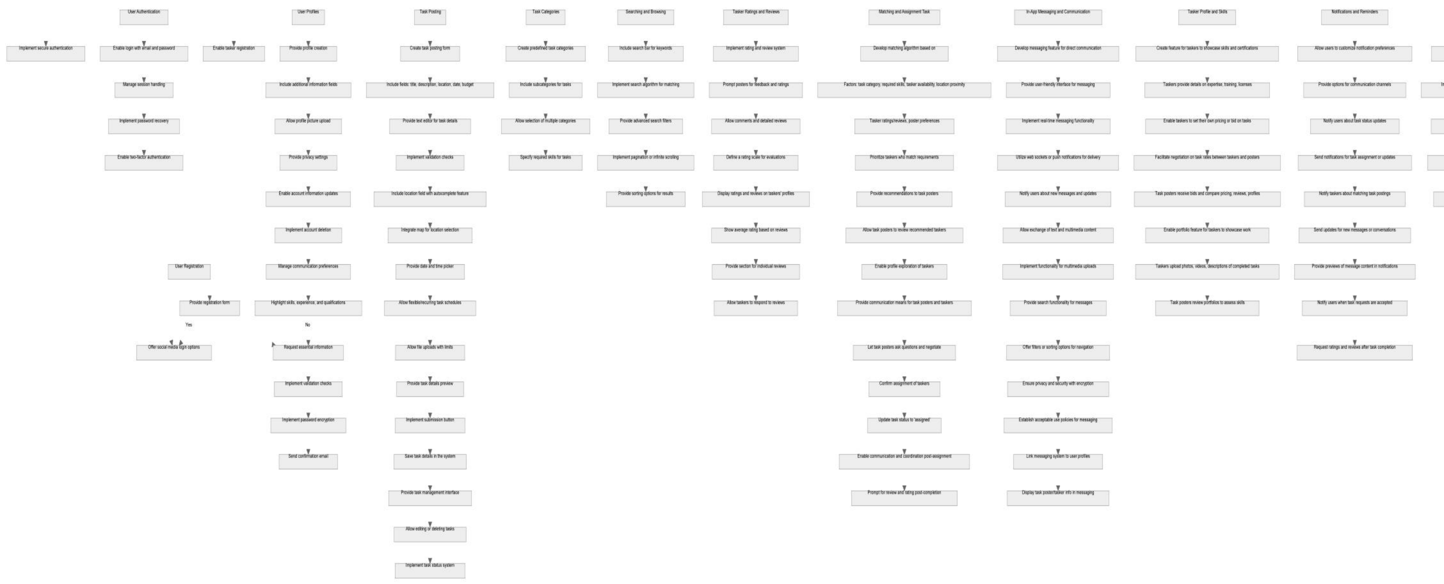
3.9 Other Requirements

Catchall section for any additional requirements.

4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable to the SRS's requirements.

4.1 Flow Chart Doagrams


User Authentication
Implement secure authentication
Enable login with email and password
Manage session handling
Implement password recovery
Enable two-factor authentication
Enable tasker registration
User Registration
Provide registration form
Yes
Offer social media login options
User Profiles
Provide profile creation
Include additional information fields
Allow profile picture upload
Provide privacy settings
Enable account information updates
Implement account deletion
Manage communication preferences
Highlight skills, experience, and qualifications
No
Request essential information
Implement validation checks
Implement password encryption
Send confirmation email
Task Posting
Create task posting form
Include fields: title, description, location, date, budget
Provide text editor for task details
Implement validation checks
Include location field with autocomplete feature
Integrate map for location selection
Provide date and time picker
Allow flexible/recurring task schedules
Allow file uploads with limits
Provide task details preview
Implement submission button
Save task details in the system
Provide task management interface
Allow editing or deleting tasks
Implement task status system
Task Categories
Create predefined task categories
Include subcategories for tasks
Allow selection of multiple categories
Specify required skills for tasks
Searching and Browsing
Include search bar for keywords
Implement search algorithm for matching
Provide advanced search filters
Implement pagination or infinite scrolling
Provide sorting options for results
Tasker Ratings and Reviews
Implement rating and review system
Prompt posters for feedback and ratings
Allow comments and detailed reviews
Define a rating scale for evaluations
Display ratings and reviews on taskers' profiles
Show average rating based on reviews
Provide section for individual reviews
Allow taskers to respond to reviews
Matching and Assignment Task
Develop matching algorithm based on
Factors: task category, required skills, tasker availability, location proximity
Tasker ratings/reviews, poster preferences
Prioritize taskers who match requirements
Provide recommendations to task posters
Allow task posters to review recommended taskers
Enable profile exploration of taskers
Provide communication means for task posters and taskers
Let task posters ask questions and negotiate
Confirm assignment of taskers
Update task status to 'assigned'
Enable communication and coordination post-assignment
Prompt for review and rating post-completion
In-App Messaging and Communication
Develop messaging feature for direct communication
Provide user-friendly interface for messaging
Implement real-time messaging functionality
Utilize web sockets or push notifications for delivery
Notify users about new messages and updates
Allow exchange of text and multimedia content
Implement functionality for multimedia uploads
Provide search functionality for messages
Offer filters or sorting options for navigation
Ensure privacy and security with encryption
Establish acceptable use policies for messaging
Link messaging system to user profiles
Display task poster/tasker info in messaging
Tasker Profile and Skills
Create feature for taskers to showcase skills and certifications
Taskers provide details on expertise, training, licenses
Enable taskers to set their own pricing or bid on tasks
Facilitate negotiation on task rates between taskers and posters
Task posters receive bids and compare pricing, reviews, profiles
Enable portfolio feature for taskers to showcase work
Taskers upload photos, videos, descriptions of completed tasks
Task posters review portfolios to assess skills
Notifications and Reminders
Allow users to customize notification preferences
Provide options for communication channels
Notify users about task status updates
Send notifications for task assignment or updates
Notify taskers about matching task postings
Send updates for new messages or conversations
Provide previews of message content in notifications
Notify users when task requests are accepted
Request ratings and reviews after task completion

4.3 Data Flow Diagrams (DFD)

4.2 State-Transition Diagrams (STD)

5. Change Management Process

Identify and describe the process used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved?