| **Software Requirement Specifications**  **Tutorly (Online Tuition Platform)**  **Version 1.0 approved**   | Project Code | Software Engineering (CS 3009) | | --- | --- | | Supervisor | Not Applicable | | Co Supervisor | Not Applicable | |  |  | | Project Team | Raahim ,Kainat,Hashir | | Submission Date |  | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Document History

Not applicable

| **Version** | **Name of Person** | **Date** | **Description of change** |
| --- | --- | --- | --- |
|  |  |  | [e.g. Document Created] |
|  |  |  | [Added Non-functional requirements] |
|  |  |  | [Added UseCase x.x.xx] |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Distribution List

Not Applicable

| **Name** | **Role** | |
| --- | --- | --- |
|  | | Supervisor |
|  | | Co- Supervisor |
|  | |  |

Document Sign-Off

Not Applicable

| **Version** | **Sign-off Authority** | **Sign-off Date** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[*1.*](#_pk52jr5apfr2) *Introduction 7*

[1.1.](#_2qatoe8psnwo) Purpose of Document 7

[1.2.](#_w38uq1lxeva) Intended Audience 7  
1.3 Abbreviations ………………………………………………………………………………………...7

[1.4.](#_np0922y6yt4i) Document Convention 7

[*2.*](#_uoo60u5vgx0u) *Overall System Description 8*

[2.1.](#_iqb5shu2ts5l) Project Background 8

[2.2.](#_tzmqtwunbfyo) Project Scope 8

[2.3.](#_o4bc9yar4dt3) Not In Scope 8

[2.4.](#_gsnnwkplem9) Project Objectives 8

[2.5.](#_j3fcz0w155oa) Stakeholders 8

[2.6.](#_4ngadriqiy05) Operating Environment 8

[2.7.](#_rue1kobbdnoo) System Constraints 8

[2.8.](#_enno9ussarxl) Assumptions & Dependencies 8

[*3.*](#_5s2mo3muze9n) *External Interface Requirements 9*

[3.1.](#_xnbqtfrz7p63) Hardware Interfaces 9

[3.2.](#_mykl46o6deou) Software Interfaces 9

[3.3.](#_2blibz23q21u) Communications Interfaces 9

[*4.*](#_mh57epy1xlzq) *Functional Requirements 10*

[*4.1.*](#_emb4qfmodwvk) *Functional Hierarchy 10*

[4.2.](#_lp019mf9e2cw) Use Cases 10

[4.2.1.](#_gt3tc5zij0fc) [Title of use case] 10

[*5.*](#_dgsd9ovwj2wl) *Non-functional Requirements 11*

[5.1.](#_od8gh8smb6b9) Performance Requirements 11

[5.2.](#_axn89r9xo1x) Safety Requirements 11

[5.3.](#_o9hi22926zt) Security Requirements 11

[5.4.](#_gukfzcya84xl) User Documentation 11

[*6.*](#_rtxuwtdwkfo4) *References 12*

[*7.*](#_qadyirhqy4h0) *Appendices 13*

1. ***Introduction***

* 1. ***Purpose of Document***

This Software Requirements Specification (SRS) document describes in detail the functional and non-functional requirements for Tutorly (Online Tuition Platform). It provides a comprehensive guide for developers, project managers, and testers to understand and implement the system.

* 1. ***Intended Audience***

| Audience | Relevance | Key Sections |
| --- | --- | --- |
| Developers | Implement system features based on functional and technical requirements. | Sections 3, 4 (Functional Requirements) |
| Project Managers | Oversee timelines, scope, and resource allocation. | Sections 1, 2 (Overview, Objectives) |
| Testers | Verify system behavior against requirements. | Sections 4 (Use Cases), 5 (Non-Functional) |
| UI/UX Designers | Design interfaces based on interaction flows. | Sections 3.2 (Software Interfaces), 4.2 |
| Stakeholders | Review alignment with business goals (e.g., tutors, students). | Sections 2 (Project Scope, Objectives) |

**1.3 Abbreviations**

| Abbreviation | Description |
| --- | --- |
| SRS | Software Requirements Specification |
| JWT | JSON Web Token |
| REST | Representational State Transfer |

* 1. ***Document Convention***

**Font:** Arial, size 10 pt.

**Headings:** Bold, Italic,hierarchical numbering.

**Requirement IDs:** REQ-, prefixed by priority (H/M/L).

1. ***Overall System Description***
   1. ***Project Background***

Academic institutions and learners need a centralized platform to connect students with qualified tutors. Tutorly addresses this need by enabling secure interactions, contract management, and performance tracking.

* 1. ***Project Scope***

The platform allows students to search for tutors by subject, language, and experience; manage contracts and payments via Stripe; submit reviews; and track tutor availability. Tutors can set availability, respond to contracts, and view feedback.

* 1. ***Not In Scope***
* Video conferencing integration
* Offline/desktop application support
  1. ***Project Objectives***
* Enable secure registration and login for students and tutors.
* Facilitate efficient tutor search and filtering.
* Automate contract creation and Stripe-based payments.
* Support review submission and dynamic rating updates.
  1. ***Stakeholders***
* Students: Seek tutoring services.
* Tutors: Offer educational services.
* Administrators: Oversee user roles and system analytics.
* Project Team: Raahim, Kainat, Hashir.
  1. ***Operating Environment***
* Client: Modern web browsers (Chrome, Firefox, Edge).
* Server: Vercel (frontend), Railway PostgreSQL (database), Serverless functions.
* Third-Party: Stripe for payment processing.
  1. ***System Constraints***
* Software Constraints: The application must run on modern web browsers and relies on specific versions of third-party services like Stripe, which must remain stable and supported. Serverless deployment models on Vercel and Railway also impose architectural constraints such as cold start times and function execution limits.
* Cultural Constraints: Since the platform targets a diverse user base, the primary language is English. Multilingual support may be considered later, but currently, it limits accessibility to English-speaking users.
* Legal Constraints: The system must comply with data protection and privacy regulations, and it should not store sensitive payment data directly, delegating this to Stripe.
* User Constraints: As students may vary in age and technical proficiency, the UI must be intuitive and lightweight. Visual aids like icons and consistent layout patterns are preferred over complex text-based interfaces.
* Off-the-Shelf Component Constraints: Integration with Stripe means that the platform inherits limitations such as region-based availability, payment method support, and rate limits. These must be accounted for in contract and payment flow design.
  1. ***Assumptions & Dependencies***
* Users have stable internet connections.
* Stripe API remains available and unchanged.
* Vercel and Railway services maintain uptime.

1. ***External Interface Requirements***
   1. ***Hardware Interfaces***

The system interacts with standard desktop devices and smartphones. No specialized hardware is required.

* 1. ***Software Interfaces***

Stripe API (v2023-10-16): Used for handling payment processing and secure transactions. When a student initiates a payment, the backend generates a Stripe checkout session. The session ID and success/cancel URLs are passed between frontend and backend. Stripe sends webhook responses to the server indicating payment status, which is then recorded in the contract database.

PostgreSQL (v16.2): Acts as the main relational database. It stores information on users (students and tutors), contracts, reviews, availability slots, and session history. Data like user credentials (hashed), tutor profiles, and feedback are accessed across backend endpoints.

Serverless Functions (Vercel): These functions expose RESTful APIs that manage authentication, CRUD operations, search queries, and payment logic. They communicate using JSON and handle business logic before reading from or writing to the database.

Nodemailer (v6.9.4): Used for sending transactional emails, such as account registration confirmations, contract updates, and review notifications.

JWT (jsonwebtoken v9.0.2): Handles session-based authentication via signed tokens shared between the backend and frontend. Tokens include encoded user roles and expiry timestamps.

Next.js (v14.0.3): The frontend framework consumes REST APIs and dynamically renders components based on user roles. It manages session tokens locally and displays real-time data from the backend.

* 1. ***Communications Interfaces***

REST API: JSON-formatted requests and responses over HTTPS.

Real Time Notifications: Notify tutors and students on contract events (e.g., new contract, review submitted).

1. ***Functional Requirements***
   1. ***Functional Hierarchy***

* Authentication & Authorization
* Tutor Search & Filtering
* Contract Management
* Payment Processing
* Review Management
* Availability Management
  1. ***Use Cases***
     1. **User Authentication**

| **Use case 1: User Authentication** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 1 | | |
| **Actors:**  Student, Tutor | | | | |
| **Feature:** Secure access to platform | | | | |
| **Pre-condition:** | | User has valid account credentials | | |
| **Scenarios** | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User navigates to login page | | | System displays the login form |
| **2.** | User submits credentials | | | System validates input, generates JWT |
| **3.** | User receives response | | | System redirects based on user role |
| **Alternate Scenarios:** | | | | |
| **2a: Invalid credentials → System shows error, prompts retry** | | | | |
| **Post Conditions** | | | | |
| **Step#** | **Description** | | | |
| **1** | Valid session started for authenticated user | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | UC-2 , UC-3 ,UC-4 | |

* + 1. **Filter Tutors**

| **Use case 2: Filter Tutors** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 2 | | |
| **Actors:**  Student | | | | |
| **Feature:** Tutor discovery | | | | |
| **Pre-condition:** | | Student is logged in | | |
| **Scenarios** | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Student accesses search page | | | System displays filters/search bar |
| **2.** | Student applies filters (e.g., subject, rating) | | | System queries database |
| **3.** | Student views results | | | Filtered tutor list displayed |
| **Alternate Scenarios:** | | | | |
| **2a: No tutors match → System shows "No results" message** | | | | |
| **Post Conditions** | | | | |
| **Step#** | **Description** | | | |
| **1** | Tutors matching criteria displayed | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | None | |

* + 1. **Set Tutor Availability**

| **Use case 3: Set Tutor Availability** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 3 | | |
| **Actors:**  Tutor | | | | |
| **Feature:** Availability management | | | | |
| **Pre-condition:** | | Tutor is logged in | | |
| **Scenarios** | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Tutor accesses calendar | | | System loads availability slots |
| **2.** | Tutor selects available times | | | Updates database |
| **Alternate Scenarios:** | | | | |
| **2a: No slots selected → System shows "Select slots" error** | | | | |
| **Post Conditions** | | | | |
| **Step#** | **Description** | | | |
| **1** | Updated availability visible to students | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | None | |

* + 1. **Submit Review**

| **Use case 4: Submit Review** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | 4 | | |
| **Actors:**  Student | | | | |
| **Feature:** Rating And reviews | | | | |
| **Pre-condition:** | | Contract is Completed | | |
| **Scenarios** | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Student opens completed contract | | | System displays review form |
| **2.** | Student submits rating/feedback | | | Saves review, updates tutor rating |
| **Alternate Scenarios:** | | | | |
| **2a: Missing rating → System prompts "Rating required"** | | | | |
| **Post Conditions** | | | | |
| **Step#** | **Description** | | | |
| **1** | Review stored; tutor rating recalculated | | | |
|  |  | | | |
|  |  | | | |
| **Use Case Cross referenced** | | | None | |

1. ***Non-functional Requirements***
   1. ***Performance Requirements***

* The system shall support up to 1,000 concurrent users with average response times of less than 2 seconds under typical load.
* Backend services should respond to REST API requests within 500ms for 95% of requests.
* System should process payment and contract confirmation workflows within 3 seconds.
  1. ***Safety Requirements***

Automatic daily backups of the PostgreSQL database must be performed and retained for at least 7 days.

* 1. ***Security Requirements***
* All data exchanges must be conducted over secure HTTPS connections.
* Passwords must be hashed using modern cryptographic algorithms (e.g., bcrypt).
* JWT tokens must have expiration periods and be securely signed with a private secret.
* Role-based access control (RBAC) must be enforced to ensure separation of user privileges.
* Sensitive data must not be stored in the frontend or in cookies.
  1. ***User Documentation***
* Detailed user manuals must be available in both PDF and online formats.
* Inline documentation, onboarding tooltips, and video tutorials should be provided for user support.

1. ***References***

Stripe API Documentation: <https://stripe.com/docs>

Vercel Deployment Guide: <https://vercel.com/docs>

Railway Documentation: <https://railway.app/docs>

1. ***Appendices***

UseCase Diagrams:  


