Software Requirements Specification

TutorFIT

A Mobile Application Connecting Students and Tutors at the Florida Institute of Technology

Project Team:
- Eleanor Barry (ebarry2022@my.fit.edu)
- Sidney Nedd (snedd2020@my.fit.edu)
- Samaher Damanhori (sdamanhori2020@my.fit.edu)

Faculty Advisor:
Dr. Khaled Salhoub (kslhoub@fit.edu)

Client:
- Dr. Khaled Salhoub
- Students of the Florida Institute of Technology

10/02/2023

FLORIDA TECH
FLORIDA'S STEM UNIVERSITY®

Contents

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for the development of the "TutorFIT" mobile application. It outlines the features, functionality, and constraints that will guide the development process.

1.2 Scope

The scope of this project includes the creation of a mobile application that connects students and tutors at the Florida Institute of Technology (FIT). The application aims to streamline the process of connecting students with tutors, offering scheduling, communication, notifications, and user engagement features.

1.3 Definitions, Acronyms, and Abbreviations

Terms Used	**Definitions**
FIT	Florida Institute of Technology
SDK	Software Development Kit
API	Application Programming Interface
Xamarin	A cross-platform app development framework
Flutter	A cross-platform app development framework

2. Overall Description

2.1 Product Perspective

The "TutorFIT" application will serve as an independent mobile application that interacts with FIT's course and user databases through web scraping. It does not have direct integration with FIT's internal systems.

2.2 Product Features

The application will provide the following key features:

- User Registration
- Scheduling
- Communication
- Push Notifications/Alerts
- Student Ratings and Reviews
- Event Tracking
- User Engagement and Retention

2.3 User Classes and Characteristics

There are two primary user classes:

2.3.1. Students: Users seeking tutoring services.

2.3.2 Tutors: Users offering tutoring services.

2.4 Operating Environment

The application will be developed for both Android and iOS platforms, ensuring compatibility with a wide range of devices.

2.5 Design and Implementation Constraints

- The application will be developed using cross-platform frameworks (e.g., Xamarin, Flutter).
- Access to FIT's API is limited; web scraping will be used to obtain course data.
- Backend server-side components will be developed using JavaScript (Node.js).
- Third-party tools and libraries will be integrated as needed.

2.6 Assumptions and Dependencies

- Users are responsible for providing accurate and up-to-date information during registration.

- The application's functionality relies on the availability and structure of FIT's web pages for data retrieval.

3. System Features

3.1 User Registration

1. **User Sign-up:** Users can register by providing their name, email address, and password.
    - Sample Input (Correct): User enters a valid email and password.
    - Sample Output (Correct): User is successfully registered and redirected to the home screen.
    - Sample Input (Incorrect): User enters an invalid email or password.
    - Sample Output (Incorrect): User receives an error message and is prompted to correct the input.

2. **User Profiles:** Users can create and edit their profiles, including personal information such as name, contact details, and a profile picture.
    - Sample Input (Correct): User fills in all required profile information.
    - Sample Output (Correct): User profile is successfully created or updated.
    - Sample Input (Incorrect): User misses required fields.
    - Sample Output (Incorrect): User is prompted to complete all required fields.

3. **Course Registration:** Users can select and register for specific courses from a predefined list.
    - Sample Input (Correct): User selects courses from the provided list.
    - Sample Output (Correct): User's selected courses are added to their profile.
    - Sample Input (Incorrect): User selects courses not available in the list.
    - Sample Output (Incorrect): User receives an error and is asked to select from the provided courses.

4. **Live Search:** The system will implement a live search feature during registration, allowing users to quickly find and select their courses, professors, and departments.
    - Sample Input (Correct): User starts typing, and relevant suggestions appear.
    - Sample Output (Correct): Users can easily select relevant courses and professors.
    - Sample Input (Incorrect): No suggestions appear as the user types.
    - Sample Output (Incorrect): User faces difficulty in selecting courses and professors.

5. **Language Preference:** Users can set their preferred teaching or learning language during registration.
   - Sample Input (Correct): User selects their preferred teaching or learning language.
   - Sample Output (Correct): Language preference is saved in the user's profile.
   - Sample Input (Incorrect): User encounters issues while setting language preference.
   - Sample Output (Incorrect): User's language preference is not saved, and they may need to retry.

3.2 Scheduling

1. **Class Enrollment:** Students can view the list of classes they are enrolled in.
   - Sample Input (Correct): Student views their enrolled classes.
   - Sample Output (Correct): Student sees a list of their enrolled classes.
   - Sample Input (Incorrect): Student's enrolled classes are not displayed.
   - Sample Output (Incorrect): Students are unable to see their classes.

2. **Tutor Selection:** Students can access information about available tutors for their classes.
   - Sample Input (Correct): Student selects a course and views available tutors.
   - Sample Output (Correct): Student sees a list of available tutors for the selected course.
   - Sample Input (Incorrect): No tutors are displayed for the selected course.
   - Sample Output (Incorrect): Student is informed that no tutors are available.

3. **Tutor Scheduling:** Students can schedule tutoring sessions with available tutors.
   - Sample Input (Correct): Student schedules a tutoring session with an available tutor.
   - Sample Output (Correct): Student receives a confirmation, and the session is added to their schedule.
   - Sample Input (Incorrect): Student encounters an error while scheduling.
   - Sample Output (Incorrect): Student receives an error message and is unable to schedule.

4. **Tutor Management:** Tutors can view the list of classes they have chosen to tutor for during registration.
   - Sample Input (Correct): Tutor views the list of classes they can tutor for.
   - Sample Output (Correct): Tutor sees the classes they can tutor for and can manage their availability.
   - Sample Input (Incorrect): Tutor cannot view their classes.
   - Sample Output (Incorrect): Tutor is unable to manage their availability.

5. **Tutor Requests:** Tutors can manage incoming tutoring requests from students, accepting or declining them as needed.
   - Sample Input (Correct): Tutor receives a tutoring request and accepts it.
   - Sample Output (Correct): Tutor confirms the appointment, and it's added to their schedule.
   - Sample Input (Incorrect): Tutor declines a tutoring request.
   - Sample Output (Incorrect): Student is notified of the decline, and the appointment is not scheduled.

6. **Search Filters:** Users can search for tutors using filters such as course name, course code, preferred learning language, and department.
   - Sample Input (Correct): User applies filters to search for tutors.
   - Sample Output (Correct): User receives a filtered list of tutors matching their criteria.
   - Sample Input (Incorrect): Filters do not work as expected.
   - Sample Output (Incorrect): User does not receive a filtered list of tutors.

3.3 Communication

1. **In-App Messaging:** Users can communicate with each other via an in-app messaging system.
   - Sample Input (Correct): Users send and receive messages in real-time.
   - Sample Output (Correct): Messages are delivered instantly between users.
   - Sample Input (Incorrect): Messages are delayed or not delivered.
   - Sample Output (Incorrect): Users experience communication issues.

2. **Real-time Chat:** Messaging should be real-time, leveraging real-time messaging SDKs to ensure immediate communication.
   - Sample Input (Correct): Users engage in a real-time chat session.
   - Sample Output (Correct): Chat messages appear in real-time.
   - Sample Input (Incorrect): Messages lag or don't appear instantly.
   - Sample Output (Incorrect): Users experience delays in messaging.

3. **Email Integration:** Users have the option to communicate through email if they prefer email-based communication.
   - Sample Input (Correct): Users choose to communicate via email.
   - Sample Output (Correct): Email communication is initiated without issues.
   - Sample Input (Incorrect): Users encounter problems when trying to communicate via email.
   - Sample Output (Incorrect): Email communication fails or does not initiate.

3.4 Push Notifications/Alerts

1. **Appointment Alerts:** Users will receive push notifications and alerts for appointment updates, including accepted and declined appointments.
   - Sample Input (Correct): Users receive push notifications for appointment updates.
   - Sample Output (Correct): Users are notified of accepted and declined appointments.
   - Sample Input (Incorrect): Push notifications for appointments are not received.
   - Sample Output (Incorrect): Users do not receive appointment updates.

2. **Reminder Alerts:** Users will receive reminders for scheduled tutoring sessions.
   - Sample Input (Correct): Users receive reminders for scheduled tutoring sessions.
   - Sample Output (Correct): Users are reminded of upcoming sessions.
   - Sample Input (Incorrect): Users do not receive session reminders.
   - Sample Output (Incorrect): Users miss scheduled sessions due to lack of reminders.

3. **Message Alerts:** Users will be notified of new messages from their tutors.
   - Sample Input (Correct): Users are alerted of new messages from their tutors.
   - Sample Output (Correct): Users receive timely alerts for new messages.
   - Sample Input (Incorrect): Message alerts are not received.
   - Sample Output (Incorrect): Users are not notified of new messages.

3.5 Student Reviews and Ratings

1. **Review Submission:** Students can submit reviews and ratings after each tutoring session.
   - Sample Input (Correct): Students can submit reviews and ratings after each tutoring session.
   - Sample Output (Correct): Reviews and ratings are successfully submitted.
   - Sample Input (Incorrect): Students encounter issues while submitting reviews.
   - Sample Output (Incorrect): Reviews and ratings are not submitted, and students may need to retry.

2. **Review Visibility:** Reviews and ratings are visible to other students considering the same tutor.
   - Sample Input (Correct): Reviews and ratings are visible to other students.
   - Sample Output (Correct): Students can see reviews and ratings when considering a tutor.

- Sample Input (Incorrect): Reviews and ratings are not visible to other students.
- Sample Output (Incorrect): Students do not see reviews and ratings.

3. **Tutor Responses:** Tutors can view and respond to student feedback.
    - Sample Input (Correct): Tutors can view and respond to student feedback.
    - Sample Output (Correct): Tutors can engage with student reviews.
    - Sample Input (Incorrect): Tutors encounter issues while responding to feedback.
    - Sample Output (Incorrect): Tutors are unable to respond to student feedback.

3.6 Event Tracking

1. **User Engagement Metrics:** The application will integrate a third-party SDK to monitor user engagement, tracking activities such as sign-ups, appointments booked, cancellations, and other relevant interactions.
    - Sample Input (Correct): The system accurately tracks user engagement activities.
    - Sample Output (Correct): Stakeholders receive data on sign-ups, appointments, and user interactions.
    - Sample Input (Incorrect): Engagement metrics are not tracked correctly.
    - Sample Output (Incorrect): Stakeholders do not receive accurate data on user interactions.

2. **Data Collection**: The system will collect data on user behavior within the app.
    - Sample Input (Correct): The system collects data on user behavior.
    - Sample Output (Correct): Data on user interactions is stored for analysis.
    - Sample Input (Incorrect): Data collection fails.
    - Sample Output (Incorrect): No data on user interactions is stored.

3.7 User Engagement and Retention

1. **Loyalty Program:** Loyal and frequent users will have the advantage of accessing free tutoring sessions.
    - Sample Input (Correct): Loyal users are rewarded with free tutoring sessions.
    - Sample Output (Correct): Loyal users receive free sessions as part of the loyalty program.
    - Sample Input (Incorrect): Loyalty program does not provide free sessions as expected.
    - Sample Output (Incorrect): Loyal users do not receive free sessions.

2. **Ad Revenue Redistribution:** Dedicated tutors will benefit from ad revenue redistribution, allowing them to earn extra income.
    - Sample Input (Correct): Tutors benefit from ad revenue redistribution.
    - Sample Output (Correct): Tutors receive additional income from ad revenue.
    - Sample Input (Incorrect): Ad revenue redistribution does not work as expected.
    - Sample Output (Incorrect): Tutors do not receive additional income.

3. **Reward System:** The app will encourage users to stay engaged and active through a reward system.
    - Sample Input (Correct): Users are incentivized to stay engaged through rewards.
    - Sample Output (Correct): Users receive rewards for engagement.
    - Sample Input (Incorrect): The reward system does not motivate user engagement.
    - Sample Output (Incorrect): Users do not receive rewards as expected.

4. External Interface Requirements

4.1 User Interfaces

The application will provide user-friendly interfaces for both students and tutors.

4.2 Hardware Interfaces

The application will require standard smartphone hardware components like cameras, microphones, and push notification capabilities.

4.3 Software Interfaces

The application will interact with third-party libraries, SDKs, and FIT's web pages for data retrieval.

5. Software Attributes

5.1 Performance

Functional Requirements:

1. **Responsiveness:** The application must respond promptly to user interactions, ensuring a seamless user experience.
2. **Scalability:** The system should be able to handle an increasing number of users and data over time without significant performance degradation.

Non-Functional Requirements:

- **Response Time:** The application should maintain low response times, even with a growing user base.
- **Reliability:** Users should experience minimal downtime, ensuring access to scheduling and communication features at all times.

5.2 Security

Functional Requirements:

1. **User Data Protection:** User data, including personal information and communication, must be securely stored and transmitted.
2. **Authentication:** Secure authentication mechanisms must be implemented to ensure user privacy and data integrity.

Non-Functional Requirements:

- **Data Encryption:** User data transmission should be encrypted to prevent unauthorized access.
- **Data Integrity:** The system should protect against data breaches and ensure the confidentiality of user information.

5.3 Reliability

Functional Requirements:

1. **System Availability:** The application must be available for scheduling, communication, and other features.
2. **Message Delivery:** Messages must be reliably delivered to ensure effective communication between users.

Non-Functional Requirements:

- **Uptime:** The application should have high uptime, with minimal disruptions to user access.
- **Message Reliability:** Messages should be delivered without loss or delay.

5.4 Usability

Functional Requirements:

1. **Intuitive Design:** The user interface must be intuitive, guiding users through registration, scheduling, and communication.
2. **Clear Notifications:** Push notifications and alerts should be clear and informative.

Non-Functional Requirements:

- **User Satisfaction:** Users should find the application easy to use and understand, promoting high user satisfaction.

5.5 Maintainability

Functional Requirements:

1. **Code Documentation:** The application codebase must be well-documented for future maintenance.
2. **Scalable Architecture:** The application architecture should support future enhancements and changes.

Non-Functional Requirements:

- **Code Quality:** Code quality and organization should facilitate maintenance and updates.

5.6 Scalability

Functional Requirements:

1. **User Growth:** The application must handle an increasing number of users without significant performance issues.
2. **Data Handling:** The system should manage a growing dataset efficiently.

Non-Functional Requirements:

- **Scalable Infrastructure:** The application's infrastructure should be capable of scaling horizontally to accommodate growth.

5.7 Privacy and Data Protection

Functional Requirements:

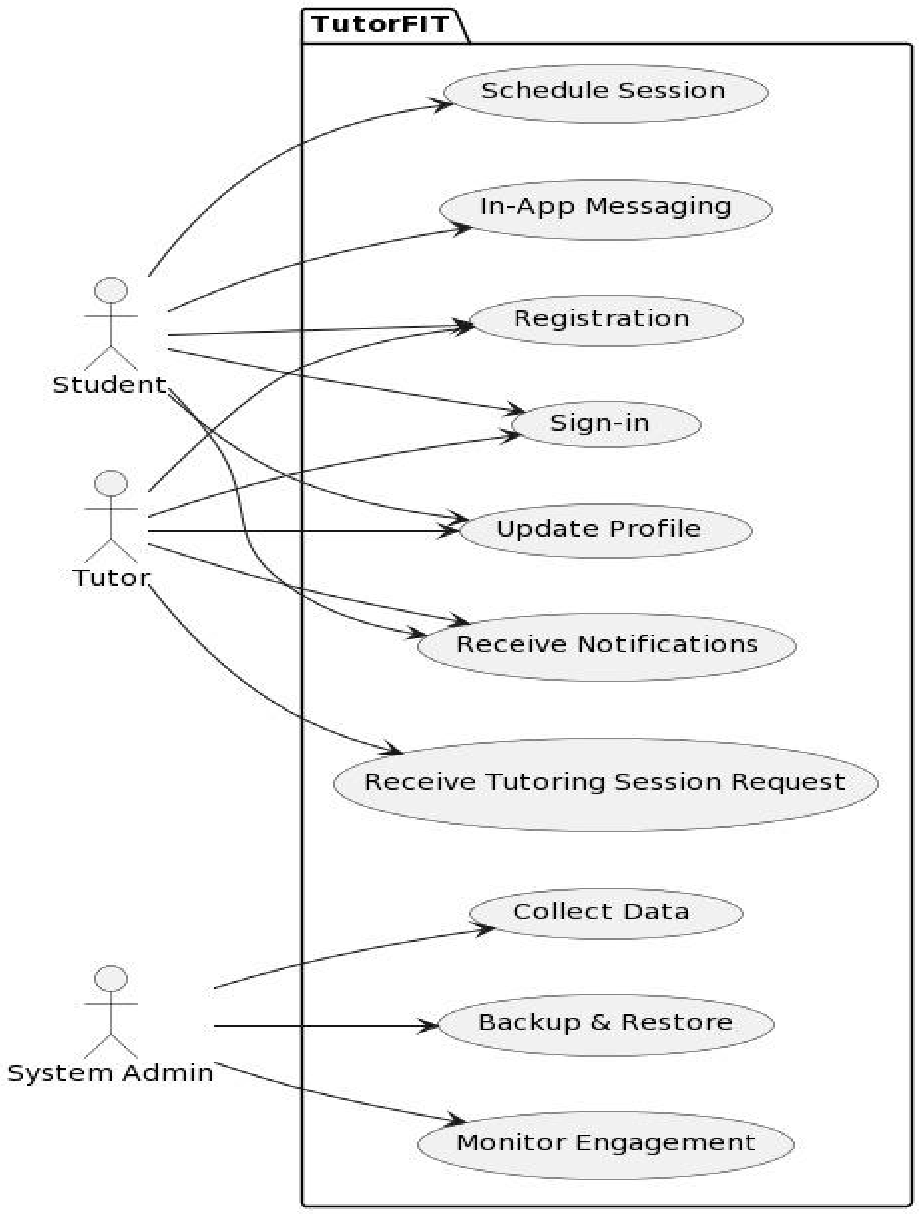
1. **Data Encryption:** User reviews, ratings, and personal data must be encrypted during transmission and storage.
2. **Data Access Control:** Access to user reviews and ratings should be restricted to authorized users.

3. **User Consent:** Users must provide consent for their reviews and ratings to be visible to others.

Non-Functional Requirements:

- **Compliance:** The system must comply with data protection regulations and guidelines to ensure the privacy and security of user data.

6. Use Case Diagram



Use case Explanation:

a. **Actors:**
   1. **Student:** Represents individuals seeking tutoring services.
   2. **Tutor:** Represents professionals providing tutoring services.
   3. **System Admin:** Represents administrative users responsible for system maintenance and oversight.

b. **Use Cases:**

   1. **Registration (UC1):** Both Students and Tutors need to register themselves in the TutorFIT system to access its features.

   2. **Sign-in (UC2):** After registration, both Students and Tutors must sign-in to access their respective dashboards and functionalities.

   3. **Update Profile (UC3):** Both Students and Tutors can update their personal and professional details in the TutorFIT system.

   4. **Receive Notifications (UC4):** Both Students and Tutors can receive notifications about various activities, such as session confirmations, reminders, or new messages.

   5. **Schedule Session (UC5):** Students can schedule tutoring sessions with available Tutors.

   6. **In-App Messaging (UC6):** Students can communicate with Tutors directly within the application using the messaging feature.

   7. **Receive Tutoring Session Request (UC7)**: Tutors will receive requests from Students who wish to schedule a tutoring session with them. They can then accept or decline these requests.

   8. **Monitor Engagement (UC8):** The System Admin has the ability to monitor user engagement on the platform, including tracking activities like sign-ups and tutoring sessions.

   9. **Collect Data (UC9):** The System Admin can initiate and manage the process of data collection related to user behavior within the app, providing insights into user preferences and behavior.

   10. **Backup & Restore (UC10):** The System Admin can back up the system data and restore it if needed. This ensures data safety and system reliability.

c. System Boundary:

The diagram has a system boundary labeled "TutorFIT", which encapsulates all the use cases mentioned. This represents the scope of the TutorFIT system and ensures that the functionality being described is within the confines of this system.

d. Relationships:

The arrows in the diagram represent the interactions between the actors and the use cases. For example, both the Student and Tutor actors interact with the "Registration" use case, indicating that both of them have the capability to register themselves on the platform.