UNIVERSITY OF ESWATINI

FACULTY OF SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

Group members:

1. Tanele Shongwe 202201117
2. Sithembiso Maseko 202201771
3. Banele Mngomezulu 202202780
4. Lusanda Dube 202200908

ONLINE TUTORING APP PROJECT PLAN

1. Introduction

Background and History

In recent years, the demand for accessible and flexible academic support has increased, particularly among university students facing diverse challenges in their educational journeys. Traditional tutoring methods are often insufficient in meeting the varying needs of learners in a fast-paced academic environment. Many students struggle to find qualified tutors who fit their schedules and learning preferences, thus leading to frustration and academic setbacks.

Aims and Objectives

The primary aim of this project is to develop a comprehensive online tutoring application that connects tertiary-level students with qualified tutors through a user-friendly platform. The objectives include:

- Designing an intuitive interface for students to find and connect with tutors easily.
- Implementing various communication methods, including video conferencing, chat, and collaborative whiteboarding tools.
- Facilitating scheduling and secure payment options for tutoring sessions.
- Allowing tutors to create detailed profiles showcasing their expertise and availability.
- Ensuring the platform is secure, efficient, and reliable.

Project Team

- **Project Leader**: Tanele Shongwe
- **Architect**: Banele Mngomezulu
- **Designer**: Lusanda Dube
- **Programmer**: Sithembiso Maseko

Project Summary

The online tutoring application aims to bridge the gap between students and tutors in a university setting, promoting collaboration and effective learning through an organized and efficient platform. By utilizing modern technology, the application seeks to empower students to achieve their academic goals and enhance their educational experiences.

2. User/Client Involvement

Information, Services, Resources, and Facilities Provided

The success of the tutoring app hinges on active participation and input from users and clients. The following resources will be made available:

- User Interviews and Surveys: Of tertiary-level students and potential tutors will be conducted to gather insights regarding their needs and preferences.
- Feedback Sessions: Regular focus groups involving users to iterate on features and gather real-time feedback during the development process.
- Access to Existing Educational Content: Clients will provide educational resources, guidelines, and materials to facilitate tutor training and app content development.
- Technical Support: IT resources, such as access to servers, databases, and necessary software throughout the development life cycle.

3. Risks

Potential Risks

Identifying and mitigating potential risks is crucial for the success of the project. The following risks have been identified:

1. **Technical Risks**
   - Scalability Issues: As user demand grows, the application's infrastructure may not support increased traffic.
   - Integration Challenges: Issues with integrating payment gateways and other third-party services could delay deployment.
   - Security Vulnerabilities: Risks of data breaches or unauthorized access to user information.
2. **Market Risks**
   - Competition: The presence of established online tutoring services may hinder user adoption rates.
   - Attracting Qualified Tutors: Difficulty in onboarding skilled tutors may lead to a lack of quality in services offered.
3. **Project Management Risks**
   - Scope Creep: Changing requirements during development phases may lead to delays and budget overruns.
   - Ineffective Communication: Lack of consistent communication among team members could lead to misunderstandings and development inefficiencies.

Mitigation Strategies

To address the identified risks, the following strategies will be implemented:

- Scalability Testing: Regular performance tests to ensure the system can handle increased loads.
- Thorough Integration Testing: Rigorous testing protocols for payment gateways and third-party integrations prior to full deployment.
- Security Protocols: Implementing secure coding practices and regular security audits to identify vulnerabilities.
- Market Analysis: Continuous evaluation of competing services to ensure unique value propositions are maintained.
- Effective Communication Framework: Use of project management tools (es.g., Jira, Trello) to keep all team members aligned on project developments and changes.

4. Standards, Guidelines, and Procedures

To ensure the success of our project, we will follow established standards, guidelines, and procedures:

- Coding Standards: adherence to industry standard coding practices such as Clean Code will be mandatory to maintain consistency and readability in our codebase.
- Agile Methodology: we will employ the Agile software development methodology, allowing for iterative development, regular feedback, and the flexibility to adapt to changing requirements throughout the project lifecycle.
- Quality Assurance: A robust quality assurance strategy, including unit testing, integration testing, and user acceptance testing (UAT), will be implemented to guarantee that the application meets high quality standards.
- User Centered Design: all user interface designs will align with usability and accessibility principles to ensure the application is easy to navigate and accessible to all users, including those with disabilities.
- Documentation: Comprehensive documentation will be created and maintained throughout the project, covering requirements , design decisions, testing results, and user manuals, to foster knowledge sharing and support future development efforts.

5. Project Organization

Project Phases and Deliverables:

- Initiation: Define project scope, objectives, budget, and timeline. Develop a detailed project plan and establish communication channels.
- Planning and Analysis: Conduct market research, gather user and client requirements, and define system functionalities. Prepare technical documentation and design specifications.
- Development and Design: Implement the system's backend and frontend functionalities, ensuring user-friendly interface and secure data management.
- Testing and Quality Assurance: Conduct rigorous testing throughout development to ensure system functionality, performance, and security. Address and resolve identified bugs or issues.

- Deployment and Training: Launch the system to users and clients, providing training and support resources. Monitor performance, gather feedback, and implement ongoing maintenance.

Team Roles and Responsibilities

1. Project Leader:
Responsibilities:

- Overall project planning and execution.
- Coordination of resources
- Risk identification and mitigation
- Tracking project progress and ensuring deliverables meet deadlines.
- Managing team conflicts and fostering a collaborative environment.

2. Architect:
Responsibilities:

- Conduct market research and gather user and client requirements.
- Analyze user needs and define system functionalities.
- Develop detailed user stories and system specifications.
- Collaborate with designers and programmers to ensure technical feasibility of features.
- Prepare test cases and participate in system testing.

3. Designer:
Responsibilities:

- Design the user interface (UI) and user experience (UX) of the system.
- Create mockups and wireframes for various system features.
- Integrate design elements with the technical capabilities of the system.
- Prepare design documentation and assets for development.

4. Programmer:
Responsibilities:

- Develop the backend and frontend of the system according to specifications.
- Implement secure data management and encryption protocols.
- Integrate various system components and ensure functionality.
- Conduct unit testing and participate in system testing.
- Document code and maintain code repository.

6. Project phases

The project will be divided into distinct phases, each with specific tasks, milestones, and estimated effort. We will adopt the Agile methodology which emphasises iterative progress and frequent reassessment.

Phase 1: Planning (week 1)
Tasks:

- Define project scope and objectives.
- Conduct market research and competitor analysis.
- Establish user and client personas.
- Develop a detailed project plan and timeline.

➢ Milestones:

- Completion of the Requirement Specification Document.
- Stakeholder approval
- Project plan approved.

Phase 2: Analysis and Design (week 2-3)
Tasks:

- UI/ UX wire framing
- Technology stack selection
- Design system architecture and data model.

➢ Milestones:

- Approval of design prototypes.
- Submission of architectural diagrams

Phase 3: Development (week 4-6)
Tasks:

- Backend and frontend development.
- API integration and data management.
- Security implementation and testing.
- Content creation and system configuration.
- Unit testing and code reviews.

➢ Milestones:

- System prototype developed.
- Core functionalities implemented and tested.
- System integration completed.

Phase 4: Testing and Quality Assurance (week 7-8)
Tasks:

- System testing and bug fixing.
- Performance and security testing.
- Data migration and validation.

➢ Milestones:

- All critical bugs fixed and system stabilized.
- Feedback from initial users

Phase 6: Deployment (week 8-9)
Tasks:

- Launch app on target platforms

- Begin promotional activities

- Milestones
    - Official launch date
    - First user tracking and feedback analysis

Phase 7: Ongoing Maintenance and Improvement (Continuous)
Tasks:

- System monitoring and performance optimization.
- Bug fixes and feature enhancements.
- Data analysis and user feedback collection.
- Regular security updates and compliance checks.

- Milestones:
- System stability and performance maintained.
- New features and improvements implemented based on user feedback.
- Security vulnerabilities addressed and system kept up-to-date.

7. Requirements Analysis and Design

Requirements Gathering:

Methods**:**

- Interviews: One-on-one or group interviews with users and stakeholders to understand their needs, pain points, and expectations.
- User observation: Observing users interacting with existing systems to identify pain points and potential improvements.
- Competitive analysis (existing apps like Khan Academy, Chegg).
- User surveys (Google Forms).

Tools and Resources:

- Interview scheduling and recording software – Zoom, voice recoder
- Google Forms (surveys).
- Jira/Trello (requirement tracking).
- Figma (wireframing)

Analysis and Modelling:

Methods:

- Use case analysis (user roles: Student, Tutor, Admin): Identifying and documenting user actions and system responses to fulfil specific goals.

- Data flow diagrams (API interactions): Representing the flow of data between different system components.
- Entity-relationship diagrams (ERDs) (database design): Defining the relationships between different data entities in the system.
- Use case diagrams(user roles: Student, Tutor, Admin)

Tools and Resources:

- Diagramming software; Lucid chart / Draw.io/ Microsoft Visio.
- API documentation; Swagger.
- UI prototypes; Figma.
- Use case management tool; Visual Paradigm
- Database modelling tool; MySQL Workbench.

Design:

Methods**:**

- Spring Boot**: MVVM** (Android) + MVC (Spring Boot).
- resource naming, HTTP methods**: RESTful** API design.
- User interface (UI) prototyping: Creating low-fidelity and high-fidelity mock-ups to visualize the system's interface and gather user feedback.
- User experience (UX) testing: Conducting usability testing with users to identify any usability issues and refine the design.
- Style guide development: Defining consistent design elements and branding guidelines for the system.

Tools and Resources:

- UI/UX design software; Figma
- UI design with XML**;** Android Studio.
- Prototyping tools; Proto.io
- API endpoint testing**;** Postman.
- Usability testing platforms; User Testing
- backend scaffolding**;** Spring Boot.

8. Implementation

These are the resources and tools needed to support implementation:

Development Frameworks:

- Backend**:** Spring Boot (Java) for the backend gives you more control, scalability, and flexibility, Hibernate (JPA).
- Frontend: Android SDK (Java/Kotlin) a comprehensive java framework widely used for cross-platform development.

Database Tools:

- Database Management Software**:** MySQL (production), H2 (testing).
- Object-Relational Mapping (ORM): Spring Data JPA or Object-Document Mapping (ODM) libraries to interact with databases more easily like ODB, an Object-Relational Mapping (ORM) solution for java language, which simplifies storing objects in databases.

UML Modelling Tool:

- Lucid chart (class diagrams, sequence diagrams).

APIs and Libraries:

- Android API calls; Retrofit
- Payment Gateways**:** Integration with services like PayPal, Skrill, or other payment APIs.
- Authentication; JWT**.**
- video calls**;** WebRTC.

Testing and Deployment Tools:

- Integration Testing**:** Postman (API testing) that integrates with GitHub repositories. It automatically builds and tests your java projects on every commit.
- Containerization: Docker for Spring Boot micro services and for creating containers to package and run the application consistently across different environments.
- Security Tools: OWASP ZAP (vulnerability scanning).

- Unit Testing: JUnit (Java) a java testing framework developed by Google that provides a framework for writing and running unit tests, Mockito (mocking).

Backup and Recovery:

- Backup Systems**:** AWS RDS automated backups and Implementing regular backups of data to prevent loss.
- Recovery Plans: Database snapshots + CI/CD rollback and other strategies and tools to recover the system in case of failures.

Data:

- User Data: Profiles, credentials (encrypted).
- Tutor Information**:** Details about tutors available for booking.

Session Data: Booking timestamps, payment logs.

9. Testing

Testing Scope: Validate functionality, security, and performance across Android and Spring Boot.

Unit Testing (Programmers)

- Objective**:** Verify individual components (e.g., login logic, API clients) which is Test individual code units (functions, modules) for functionality and accuracy.
- Tools: Junit unit testing framework, Mockito.
- Responsibilities**:**
  - Programmers write and run unit tests for their own code.
  - Fix logic errors.
  - Identify and fix bugs at code level

Integration Testing (Analysts + Programmers)

- Objective**:** Test API ↔ Android app interactions, which is how different modules of the system interact with each other.
- Tools**:** Postman, Android Espresso.
- Responsibilities**:**
  - Analysts design and document test cases for integration scenarios and ensure endpoints return correct data.

    - Programmers implement and execute integration test cases and Debug HTTP errors (e.g., 400/500 responses).
    - Debug and fix any integration issues.

Functional Testing (Analysts + Designers + Programmers)

- Objective**:** Verify if the system functions meet user requirements and specifications. Validate user flows (for example; booking → payment → chat).
- Tools: Manual testing, Selenium (web views).
- Responsibilities:
    - Analysts define user stories and test cases based on requirements.
    - Designers review test cases for UI/UX completeness and clarity.
    - Programmers implement test cases and report bugs.

Performance Testing (Programmers + Manager)

- Objective: Assess scalability (e.g., 100+ concurrent users), and evaluate system performance under load and identify bottlenecks.
- Tools**:** JMeter (API load testing), Firebase Test Lab (Android).
- Responsibilities**:**
    - Optimize database queries.
    - Programmers configure and execute performance tests.
    - Manager analyses results and makes resource allocation decisions.
    - Monitor memory leaks in Android.

Communication and Reporting:

- All team members communicate testing results, bugs, and progress through regular meetings and reporting tools such as using GitHub Issues.
- A centralized bug-tracking system is used to manage and prioritize bug fixes.

Testing Success Criteria:

- All critical functionalities pass all relevant test cases.
- Bug fixes are implemented and regression testing confirms no regressions (zero critical bugs e.g., crashes, security flaws).
- Performance meets user expectations and scalability requirements.
- Security vulnerabilities are identified and addressed to acceptable levels.
- User feedback confirms a positive and intuitive user experience.

10. Resources

This project will require both human and technical resources. Our group consists of four team members who will work collaboratively on all parts of the app development. The key resources are as follows:

Human Resources

Our team is composed of four key members, each bringing specialized skills to ensure a successful development process:

- **Project Architect**: Responsible for the overall structure and design of the application, ensuring all components work harmoniously together. The architect will also guide the technical direction and make high-level design decisions.
- **Programmer**: Focused on writing, testing, and debugging the application code. The programmer will implement the technical features and functionalities as per the design specifications.
- **Designer**: Tasked with creating an engaging and user-friendly interface. The designer will ensure the aesthetic aspects of the app are appealing and intuitive for users, utilizing tools such as Figma or Canva.
- **Project Manager**: Responsible for overseeing the project timeline, resource allocation, and team coordination. The project manager will facilitate communication among team members and ensure that milestones are met consistently.

Technical Resources

To support our development efforts, we will utilize the following technical resources:

- **Computing Devices**: Laptops/Desktops for coding, testing, and running the app to ensure a productive development workspace.
- **Internet Access**: Reliable internet access is vital for research, communication, and utilization of online development tools.

Software Tools

We will adopt the following software tools to streamline development and ensure collaboration:

- **Integrated Development Environment (IDE)**: Visual Studio Code for writing and editing code, allowing for efficient programming.
- **Database Management System**: Firebase or MySQL for robust storage of user and tutor information, ensuring data security and retrieval.
- **Version Control System**: GitHub will be utilized for collaboration, allowing team members to manage code changes and track project progress effectively.

- **Design Tools**: Figma or Canva will be employed for user interface design, helping us create an engaging and user-friendly app experience.

11. Quality Assurance

Ensuring the quality of the final product is a priority. Our quality assurance process will include:

- Code Reviews: Team members will conduct regular code reviews, providing feedback to identify errors and suggest improvements.
- Testing Protocols: We will implement comprehensive testing strategies, including:
  - Functional Testing: Verifying that all features (e.g., tutor search, booking) operate as intended.
  - User Interface (UI) Testing: Ensuring that the app's design is intuitive and visually appealing to enhance user experience.
- Bug Tracking and Fixes: Identifying and resolving bugs and issues before the final app launch.
- User Feedback: Engaging a select group of users to test the app and provide feedback for continuous improvement.

12. Changes

Recognizing that adaptability is key to project success, we will manage changes through a structured process:

- **Change Request**: Any team member wishing to propose a change will present the idea to the group.
- **Impact Review**: The team will collectively evaluate the implications of the proposed change on project timelines, costs, and overall objectives.
- **Approval Process**: Only changes that receive majority team agreement and demonstrate clear value will be implemented.
- **Documentation**: All changes will be documented to maintain transparency and enable traceability.

We understand the importance of flexibility but will aim to stick to the original plan unless a change significantly improves the app.