**PROJECT NAME: LUPA**

**GROUP MEMBERS:** Ataberk YAYLA, Can ÖZER, Emirhan KÖKSAL, İrem ÖZKAN, Onur ERGÜDEN, Zuhal Naz CANSU

|  |  |
| --- | --- |
| # | NECESSARY NEEDS FROM THE ORGANIZATIONAL PROCESS |
| 1 | Continuity of the project without any disruption in case of any changes or innovations in the development team. |
| 2 | The ability of team members in the development team to communicate instantly and intervene in case of any disruption or issue |
| 3 | Maintaining transparency and visibility into project progress. |
| 4 | The ability to quickly adapt to request changes in the project in the later stages |
| 5 | Understand the customer’s expectation from the project and their working area. Software process plans and project objectives can be changed by expectations the customers. |
| 6 | The application needs to be secure to store student data and ensure user privacy. |
| 7 | A robust data storage system is required to handle the large amount of student and course information and also that tools to analyze student data and course information to generate insights and personalized recommendations necessary. |
| 8 | Data acquisition of the historical student data, course information, and academic resources are required for the execution of the desired outcomes. |
| 9 | Personalize and maintain the deep learning model for implementation of the system. |
| 10 | The tools and budget for the software development should be determined based on requirements of the project with the unity of all the Scrum members. |

|  |
| --- |
| SOFTWARE PROCESS NAME: SCRUM |
| SOFTWARE PROCESS DESCRIPTION: |
| Scrum is an Agile software development framework and is often used in software development processes. It emphasizes iterative and incremental development, frequent inspection and adaptation, and collaboration among team members. Scrum encourages collaboration and provides flexibility because requirements and priorities can be changed between sprints. Scrum enables rapid development and adaptation of the product so that it can be continuously improved by taking into account customer feedback. This method can help manage projects more efficiently and effectively.  Scrum roles include Product Owner, Scrum Master, and Development Team. The Product Owner prioritizes the product backlog, which contains all requirements, features, and enhancements. The Scrum Master facilitates the Scrum process and removes any impediments that hinder the team's progress. The Development Team self-organizes to deliver the committed work items by the end of each sprint.  Daily scrum, sprint planning, sprint review, and sprint retrospective are key scrum events that ensure alignment, transparency, and continuous improvement.   * Sprint Planning: It's a part where the team discusses three topics: Why this sprint is valuable, what can be done this sprint, and how the chosen work will be done so that the sprint is resolved well. * Daily Scrum: During daily meetings, the team inspects the process every day, reducing complexity and improving communication. * Sprint Review: During the sprint review, stakeholders can see what the team has accomplished and what they should do or not do in the future. * Scrum Retrospective: With this meeting, the scrum team discusses how the last sprint went, the problems they faced, and how they dealt with them. The team identifies the most effective changes during this meeting. |
| SOFTWARE PROCESS MODEL: AGILE |
| Sprint 1: Data Collection and Cleaning (5 weeks (1 + 1 + 2 + 1))  Task 1: Set up data collection mechanisms to gather student information.  Task 2: Design data storage architecture to efficiently manage student data.  Task 3: Implement data cleaning algorithms to handle missing values and outliers.  Task 4: Conduct testing to ensure the accuracy and reliability of collected data.  Sprint 2: Course Information Management (6 weeks (1 + 2 + 2 + 1))  Task 1: Create a database schema to store comprehensive course details.  Task 2: Develop user interfaces for instructors to input course content, evaluation criteria, and feedback.  Task 3: Implement backend functionality to store and retrieve course information.  Task 4: Conduct integration testing to verify the seamless functioning of the course information management system.  Sprint 3: Course Review and Rating System (6 weeks (1 + 2 + 2 + 1))  Task 1: Design user interfaces for students to access course reviews and ratings.  Task 2: Develop backend services for storing and retrieving course reviews and ratings.  Task 3: Implement filtering options based on course attributes and student preferences.  Task 4: Conduct usability testing to ensure an intuitive user experience.  Sprint 4: Academic Advising Services (5 weeks (1 + 1 + 2 + 1))  Task 1: Define the requirements for academic advising services.  Task 2: Develop user roles and permissions, distinguishing between students, advisors, and administrators.  Task 3: Implement features allowing advisors to assist students in course selection.  Task 4: Conduct user acceptance testing to gather feedback from advisors and students.  Sprint 5: User Role Verification (3 weeks (1 + 1 + 1))  Task 1: Implement authentication mechanisms to verify user roles during login.  Task 2: Develop role-based access controls to restrict functionalities based on user roles.  Task 3: Conduct security testing to ensure the integrity of user role verification mechanisms.  Sprint 6: Personalized Roadmap Recommendation (6 weeks (2 +1 + 2 + 1))  Task 1: Design algorithms to generate personalized roadmaps based on student profiles and interests.  Task 2: Integrate reading materials and online articles with varying difficulty levels into the system.  Task 3: Implement recommendation features allowing students to explore suggested resources.  Task 4: Conduct performance testing to evaluate the efficiency of personalized roadmap recommendations.  Sprint 7: Personalized Elective Course Recommendations (6 weeks (1 + 2 + 2 + 1))  Task 1: Develop interfaces for students to input their interests and self-development areas.  Task 2: Design algorithms to analyze student inputs and recommend personalized elective courses.  Task 3: Implement features for students to explore and enroll in recommended elective courses.  Task 4: Conduct usability testing to gather feedback on the effectiveness of elective course recommendations.  Sprint 8: Q&A Platform Implementation (6 weeks (1 + 2 + 2 + 1))  Task 1: Design user interfaces for students to ask questions and receive guidance.  Task 2: Develop backend services for storing and retrieving questions and answers.  Task 3: Implement features for upvoting, downvoting, and rating questions and answers.  Task 4: Conduct stress testing to ensure scalability and reliability of the Q&A platform.  Sprint 9: Most Visited Questions and Filtering ( 4 weeks (1 + 1 + 1 + 1))  Task 1: Implement features allowing users to access most visited questions.  Task 2: Develop filtering options for users to refine their search based on their interests.  Task 3: Enhance user interfaces to provide a seamless browsing experience.  Task 4: Conduct regression testing to ensure the stability of implemented features. |
| REASONS TO CHOOSE THIS MODEL: |
| * Transparency: With transparency all team members can see what the others are doing and can inspect each other. With that, the probability of developers making unavoidable mistakes is minimized. * Rapid Prototyping and Feedback: Agile allows for the development of application in iterations, with frequent releases of functional features. This enables early user feedback from students, advisors, and faculty. This feedback can be incorporated into subsequent iterations, ensuring LUPA remains aligned with user needs and expectations. * Fast Adaptation to Change: The needs of students, faculty, and the university can evolve throughout development. Agile's flexible nature allows the LUPA development team to adapt to these changes efficiently. New features or modifications can be prioritized and incorporated into upcoming iterations. * Flexibility to Academic Changes: Scrum’s flexible nature allows for quick adaptation of the application for the changes in curriculum, academic requirements, or student preferences, ensuring that the application remains relevant and useful over time. * Risk Mitigation in Academic Pursuits: Incremental updates and continuous feedback in LUPA reduce the risk of academic setbacks by enabling students to identify and address challenges early in their academic pursuits. * Focus on Working Software: Agile prioritizes delivering functional features over extensive documentation. This aligns well with LUPA's goal of providing a practical application to students. A working LUPA with core functionalities can be delivered sooner, with more advanced features added later. * Improved Team Collaboration: Agile emphasizes collaboration between developers, advisors, and potential users. This fosters a sense of ownership and improves the overall quality of the LUPA application. |