Certainly! Let’s tailor the response to align more closely with **Scrum** characteristics for the educational institution’s software project:

1. **Software Development Methodology (Scrum)**:
   * **Justification**:
     + **Iterative Approach**: Scrum promotes an iterative development process with time-bound iterations called **sprints**. Each sprint delivers a potentially shippable increment of the software.
     + **Collaboration and Communication**: Scrum emphasizes close collaboration between the development team, product owner, and stakeholders. Regular ceremonies (such as sprint planning, daily stand-ups, and sprint reviews) facilitate communication.
     + **Flexibility**: Scrum allows for changes in requirements during the project, making it suitable for an upgrade to an existing system.
     + **Efficiency**: By focusing on delivering valuable features early and continuously, Scrum minimizes waste.
   * **Activities Involved in Scrum Development Process**:
     + **Product Backlog Creation**: Identify and prioritize features based on customer requirements. The product backlog serves as the source of work for the project.
     + **Sprint Planning**: At the beginning of each sprint, the team selects a subset of items from the product backlog to work on during the sprint.
     + **Development Sprints**: Developers work on implementing features, aiming to complete them within the sprint duration (usually 1-4 weeks).
     + **Daily Stand-ups**: Short daily meetings where team members discuss progress, address obstacles, and synchronize their work.
     + **Sprint Review**: At the end of each sprint, the team demonstrates completed features to stakeholders.
     + **Sprint Retrospective**: Reflect on the sprint, identify areas for improvement, and adapt processes.
2. **Product Backlog Creation**:
   * The product backlog is created initially, listing all desired features, enhancements, and bug fixes.
   * Prioritization is based on customer requirements and business value.
3. **Sprint Planning**:
   * At the beginning of each sprint, the Scrum team (including the product owner) selects a subset of items from the product backlog.
   * These selected items become the sprint backlog for that specific sprint.
4. **Development Sprints**:
   * During the sprint, the development team works on implementing the features and tasks from the sprint backlog.
   * The goal is to complete these items within the sprint duration (usually 1-4 weeks).
5. **Daily Stand-ups (Daily Scrum)**:
   * Short daily meetings (usually 15 minutes) where team members share progress, discuss obstacles, and synchronize their work.
   * Each team member answers three questions: What did I do yesterday? What will I do today? Are there any impediments?
6. **Sprint Review**:
   * At the end of each sprint, the team demonstrates the completed features to stakeholders (product owner, users, etc.).
   * Feedback is gathered, and any necessary adjustments are made.
7. **Sprint Retrospective**:
   * After the sprint review, the team holds a retrospective meeting.
   * The team reflects on the sprint, identifies areas for improvement, and adapts their processes accordingly.

Remember that Scrum emphasizes continuous improvement, and each sprint builds upon the previous one. The correct order ensures effective collaboration and successful delivery of increments.

* + **Strengths of Scrum**:
    - **Adaptability**: Scrum responds well to changing requirements, allowing adjustments during the project.
    - **Customer Involvement**: Regular feedback from the customer ensures alignment with their needs.
    - **Early Delivery**: Features are delivered incrementally, providing value sooner.
    - **Collaboration**: Cross-functional teams collaborate closely, fostering creativity and problem-solving.
  + **Weaknesses of Scrum**:
    - **Documentation**: Scrum may lack comprehensive documentation, so maintaining relevant records is essential.
    - **Scope Creep**: Frequent changes can lead to scope expansion if not managed effectively.
    - **Dependency on Customer Availability**: Active customer participation is crucial for successful Scrum projects.

1. **Design Review**:
   * When reviewing the software’s scope and user interface design, consider the following principles:
     + **Visibility**: Ensure essential functions and information are visible without overwhelming the user.
     + **Feedback**: Provide immediate feedback for user actions (e.g., button clicks, form submissions).
     + **Consistency**: Maintain a consistent layout, fonts, colors, and terminology.
     + **Error Prevention**: Design interfaces to prevent errors or guide users away from mistakes.
     + **Efficiency**: Optimize workflows to minimize user effort.
     + **Aesthetics**: Strive for an appealing and user-friendly design.
   * While I cannot provide screenshots directly, consider creating mockups or wireframes to visualize these design aspects.
2. **Software Sustainability & Improvement**:
   * **Sustainability**:
     + Regular updates, bug fixes, and security patches are essential for long-term software sustainability.
     + Consider using version control systems and automated testing to ensure stability.
   * **Areas of Improvement**:
     + **Process Methodologies**:
       - Implement continuous integration and continuous delivery (CI/CD) pipelines for efficient deployment.
       - Regularly assess technical debt and allocate time for refactoring.
     + **Design Characteristics**:
       - Enhance user experience by refining the UI based on user feedback.
       - Optimize database queries and improve performance.
       - Consider modular architecture for scalability.

Remember that Scrum encourages continuous improvement, and adapting to feedback ensures successful software development.

Certainly! Let’s address each of your points:

1. **Software Development Methodology**: For this educational institution’s software project, I recommend adopting the **Agile software development methodology**. Here’s why:
   * **Justification**:
     + **Iterative Approach**: Agile emphasizes iterative development, allowing for continuous feedback and adaptation. Given that the customer knows their requirements well, an iterative approach will facilitate regular check-ins and adjustments.
     + **Collaboration and Communication**: Agile promotes close collaboration between developers, stakeholders, and end-users. Since this project is an upgrade to an existing system, ongoing communication is crucial.
     + **Flexibility**: Agile accommodates changes in requirements, which is essential when enhancing an existing system.
     + **Efficiency**: Agile minimizes waste by focusing on delivering valuable features incrementally.
   * **Activities Involved in Agile Development Process**:
     + **Backlog Creation**: Identify and prioritize features based on customer requirements.
     + **Sprint Planning**: Break down features into smaller tasks for each sprint (typically 1-4 weeks).
     + **Development Sprints**: Developers work on implementing features.
     + **Daily Stand-ups**: Short daily meetings to discuss progress and address any obstacles.
     + **Sprint Review**: Demo completed features to stakeholders.
     + **Sprint Retrospective**: Reflect on the sprint and identify areas for improvement.
   * **Strengths**:
     + **Adaptability**: Agile responds well to changing requirements.
     + **Customer Involvement**: Regular feedback from the customer ensures alignment.
     + **Early Delivery**: Features are delivered incrementally.
     + **Collaboration**: Cross-functional teams work closely together.
   * **Weaknesses**:
     + **Documentation**: Agile may lack comprehensive documentation.
     + **Scope Creep**: Frequent changes can lead to scope expansion.
     + **Dependency on Customer Availability**: Requires active customer participation.
2. **Design Review**: When reviewing the software’s scope and user interface design, consider the following golden rules of UI design:
   * **Visibility**: Ensure that essential functions and information are visible without overwhelming the user.
   * **Feedback**: Provide immediate feedback for user actions (e.g., button clicks, form submissions).
   * **Consistency**: Maintain consistency in layout, fonts, colors, and terminology.
   * **Error Prevention**: Design interfaces to prevent errors or guide users away from mistakes.
   * **Efficiency**: Optimize workflows to minimize user effort.
   * **Aesthetics**: Strive for an appealing and user-friendly design.

Screenshots can illustrate how these principles are applied in the software’s interface. Unfortunately, I cannot provide screenshots directly, but you can create mockups or wireframes to visualize these design aspects.

1. **Software Sustainability & Improvement**:
   * **Sustainability**:
     + The software’s sustainability depends on factors like maintainability, scalability, and adaptability.
     + Regular updates, bug fixes, and security patches are essential for long-term sustainability.
     + Consider using version control systems and automated testing to ensure stability.
   * **Areas of Improvement**:
     + **Process Methodologies**:
       - Implement continuous integration and continuous delivery (CI/CD) pipelines.
       - Regularly assess technical debt and allocate time for refactoring.
     + **Design Characteristics**:
       - Enhance user experience by refining the UI based on user feedback.
       - Optimize database queries and improve performance.
       - Consider modular architecture for scalability.

Remember that software development is an ongoing process, and continuous improvement is key to sustaining and enhancing the software over time.

**\*\*disadvantages\*\***

**3. Offer Informative Feedback**

If your users have performed or are performing actions on your website, it is best to display feedback immediately so that they have an idea of where their processes are

Next, we'll take an example with a tool that many professionals use, Google Drive:

* For this example, you will have to create a new folder that you can delete right after.
* When you are in this folder, choose a document of your choice, a photo, a video, a PDF file and use drag and drop to transfer it to your folder.
* Once your transfer is in progress, this window will appear to keep you informed of the progress of your transfer.

**7. Support Internal Locus of Control**

We need to give control and freedom to the users, so that they can feel that they are in control of the system themselves, giving them some form of free will helps to reassure the user.

For example, in the screenshot below we are on Capian's YouTube page, and we can choose to turn on notifications or not, YouTube lets us choose how much notification we want to see, by doing this, YouTube is giving freedom to the users and making them feel like they have control over what they will receive.

**\*\*advantages\*\***

**1. Strive for Consistency**

Whether it is the layout, the size of the button, the color code or the tone used when writing the page, it is important to be consistent throughout the site. This consistency will allow you to develop your identity and not lose users as they navigate your site.

In the example below, you can recognize Amazon, by applying this first heuristic, there is a consistency between all the pages present on Amazon, the layout is the same, the size of the button is always similar, and the color code are very specific to Amazon.

**2. Enable Frequent Users to Use Shortcuts**

Allow your users to access all parts of the website with a minimum of clicks. To do this, you not only need to establish a good hierarchy in the menu, but also make things clear.

You should think about incorporating features for advanced users and for beginners for example, with keyboard shortcuts or macro facilities, as for example with Canva which allows its users to use shortcuts to copy and paste.

**4. Design Dialog to Yield Closure**

Remember to close any interaction made with a user based on the cause of the interaction :

* Thank you message.
* Validation message.
* Summary message during a purchase.

Your user must see the path in his action, by offering him the end of an interaction through feedback you reduce his mental load and improve his experience on your interface.

**5. Offer Simple Error Handling**

A good interface should be designed to avoid errors as much as possible. However, if something goes wrong, your system should make it easy for users to understand and resolve the problem. Simple ways to deal with errors include displaying clear error notifications and descriptive hints to resolve the problem.

For example, on our site, we tell you when you log in if your login information is wrong.

**8. Reduce Short-Term Memory Load**

The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.

Avoid overloading your site or application with information of the same level. You have to deduce which ones should be placed first or you will lose the user's attention. No matter where you place your site, whether it's the home page or the menus, make sure that your user can't be distracted by unnecessary information.

For example, Capian's home page below is very minimalist and allows the user's attention to be focused on the "call to action", most of the information we want to promote is on the page without any other information that could disrupt the user's navigation.

Given the proposed software development process utilizing Scrum methodology, the software's sustainability in the long run appears promising, although certain factors need consideration to mitigate risks of potential failure.

Scrum's iterative and adaptive approach fosters continuous improvement and responsiveness to evolving requirements and user feedback, enhancing the software's ability to meet changing needs over time (Schwaber & Sutherland, 2017). By delivering working increments of the software at the end of each sprint, Scrum ensures that stakeholders have opportunities to provide input and steer the direction of development, increasing the likelihood of producing a product that remains relevant and valuable in the long term (Babar & Chauhan, 2020).

Furthermore, Scrum's emphasis on collaboration, transparency, and customer involvement promotes stakeholder engagement and buy-in, fostering a sense of ownership and support for the software (Schwaber & Sutherland, 2017). Active participation from stakeholders, including the educational institution and end-users, facilitates knowledge transfer, ensures alignment with organizational goals, and enhances the software's sustainability by fostering a shared understanding of its purpose and functionality (Babar & Chauhan, 2020).

However, there are potential risks to the software's sustainability that must be addressed. One concern is the challenge of maintaining comprehensive documentation within the Scrum framework. While Scrum prioritizes delivering working software over extensive documentation, insufficient documentation practices can hinder the software's maintainability and scalability over time (Babar & Chauhan, 2020). Without adequate documentation, future developers may face difficulties understanding the software's architecture, design decisions, and implementation details, increasing the risk of technical debt and system complexity (Schwaber & Sutherland, 2017).

To mitigate this risk and ensure the software's long-term sustainability, teams should adopt agile documentation practices that strike a balance between agility and documentation completeness (Schwaber & Sutherland, 2017). This includes capturing essential information and knowledge artifacts throughout the development process, leveraging automated documentation tools and version control systems, and conducting regular knowledge sharing sessions and documentation reviews (Babar & Chauhan, 2020). By integrating documentation tasks into sprint activities and establishing clear standards for documenting key aspects of the software, teams can enhance its maintainability and supportability, reducing the risk of failure and ensuring its continued success in the long run.