Group 15

SE HW2 - Scrumban

I. System and Functionalities:

Tracking (Profile)

- Users can view their own profile to see posts, followers, following, bio, etc
- View profiles of other users to see posts, followers, following, bio, etc
- Monitor own activity / other users activity

Posting

- Users can create and share their own content by uploading photos or videos from their device and pin a caption to it

Commenting

- Leave comment on posts shared by other users
- Comments allow interaction and engagement with content posted
- Users can reply to comments

Sharing

- Users can share post they like with followers via chatting

Liking

- Users can express satisfaction by liking a post
- Number of likes a post receives contributes to visibility and popularity

Searching

- Users can search for specific users, hashtags, locations, topics
- Search feature allow users to discover new content / accounts based on interest

Follow & Friends

- Users can follow other users to subscribe to their updates and see their posts in their feed

- Users can also unfollow accounts if they no longer wish to see conten

Chatting

- Users can send private messages to other users

Advertisement

- Advertising options for businesses to promote their products/services
- Advertisement appear in user feeds

Verified Account (Blue Tick)

- Blue verification badge indicates account is authentic and credible
- Acts as identification of legitimate account instead of impostors

Feeds

- Feed is main section of the social media where users see posts from accounts they follow

II. Break down the requirements into tasks:

Tracking (Profile)

Frontend Development:

- Implement the UI for displaying user activity and statistics.
- Implement user profile view for own profile and others'

Backend Development:

- Develop APIs for retrieving user profile data.
- Implement logic for tracking and recording user activities (posts, likes, comments).

UI Design:

- Create intuitive and responsive designs for user profiles and activity feeds.

Posting

Frontend Development:

- Design the interface for creating and uploading posts.
- Implement features for users to attach captions and select photos or videos from their device.

Backend Development:

- Develop APIs for post creation and media upload.
- Implement storage solutions for media files.

UI Design:

- Ensure the posting interface is user-friendly and accessible across devices.

Commenting

Frontend Development:

- Design and implement UI for leaving comments and replying to comments

Backend Development:

- Develop APIs for creating, retrieving, and managing comments.

UI Design:

- Design a clean and engaging commenting interface.

Sharing

Frontend Development:

- Create a sharing interface within the chat or as a separate feature.

Backend Development:

- Develop the functionality to share posts directly with other users or groups.

UI Design:

- Design an intuitive sharing mechanism.

Liking

Frontend Development:

- Implement like button functionality

Backend Development:

- Develop APIs to manage like counts and interactions.

UI Design:

- Integrate like buttons seamlessly within posts.

Searching

Frontend Development:

- Design and implement the search interface.

Backend Development:

- Develop search algorithms and API endpoints for user, hashtag, location, and topic searches

UI Design:

- Create a responsive and efficient search experience.

Follow & Friends

Frontend Development:

- Design UI for following/unfollowing users.

Backend Development:

- Implement follow/unfollow logic and manage user subscriptions.

UI Design:

- Design follow/unfollow buttons and friends list UI

Chatting

Frontend Development:

- Implement chat interface for sending private messages

Backend Development:

- Develop messaging system with real-time updates
- Create API endpoints for sending and receiving messages

UI Design:

- Design chat interface with message input, conversation view, etc.

Advertisement

Frontend Development:

- Design and implement UI elements for displaying ads in the feed

Backend Development:

- Develop ad serving system with targeting options

UI Design:

- Design ad placement within the feed UI

Verified Account (Blue Tick)

Backend Development:

- Develop verification system and criteria
- Implement badge display on verified accounts

UI Design:

- Design badge display on verified accounts

Feeds

Frontend Development:

- Design and implement UI for displaying user feeds

Backend Development:

- Develop feed generation algorithms based on followed accounts

UI Design:

- Design feed layout and post display components

These tasks will require coordination between different teams (Backend, FrontEnd, and UI/UX Design) to ensure each feature is implemented effectively and works perfectly within the larger platform.

III. Designing Scrumban Board:

Reasons for Selecting Notion:

1. Versatility:

   Notion provides a wide range of functionalities, not just project management, including note-taking, document collaboration, and database management. This versatility makes our team to use their tools and workflows into a single platform, reducing complexity and improving efficiency.

2. Customizability:

   Notion also provides customization options, allowing our team to modify the workspace and Kanban board to match our specific needs. From layout and structure to color labels, columns, and rows, our team can configure Notion to match our workstation.

3. Collaboration:

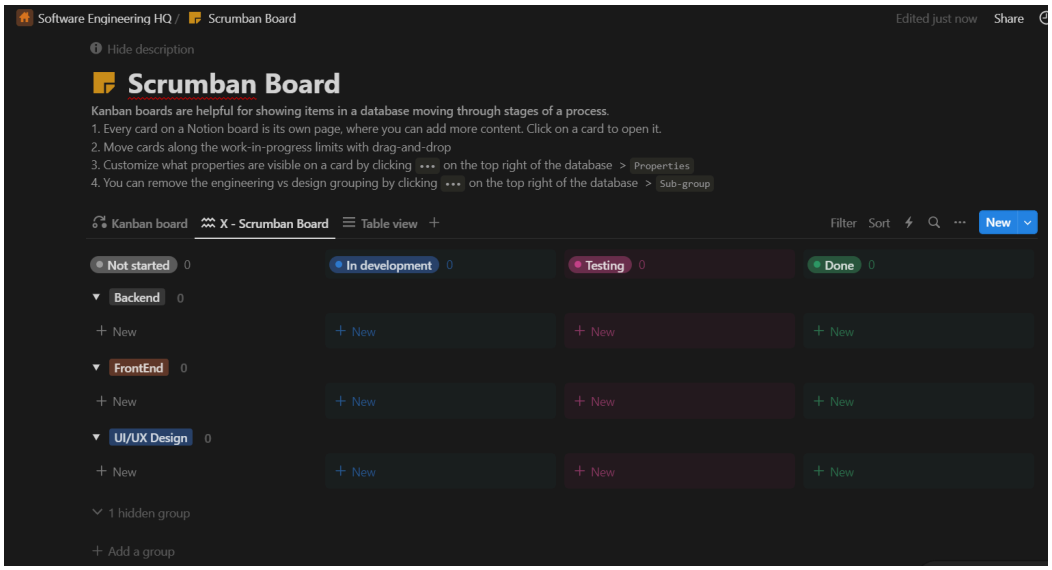
   Notion also supports collaboration among team members. With real-time editing, commenting, and mention features, team members can easily communicate and coordinate tasks directly within the platform.

4. Ease of Use:

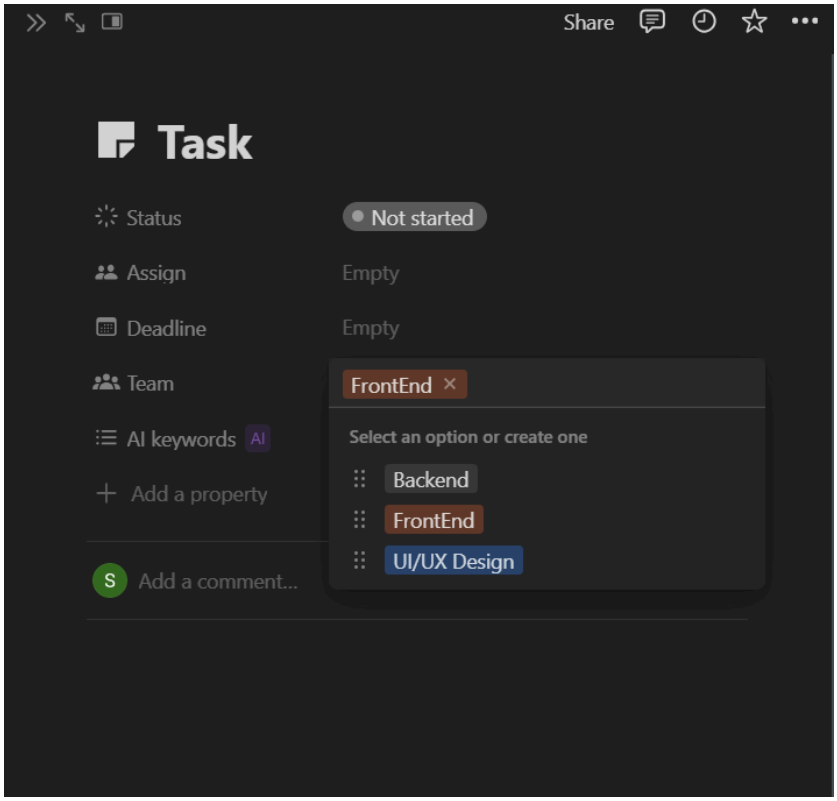
   Notion also have a user-friendly interface and intuitive design, making it accessible to team members. The drag-and-drop functionality is really helpful, also nice editing, and simple navigation that also streamline the user experience, reducing the learning curve for new users.

Scrumban Board Template:

There are four columns of the Scrumban Board: *Not Started*, *In Development*, *Testing*, and *Done*. We divided the teams into 3 group which are Backend, FrontEnd and UI/UX Design.



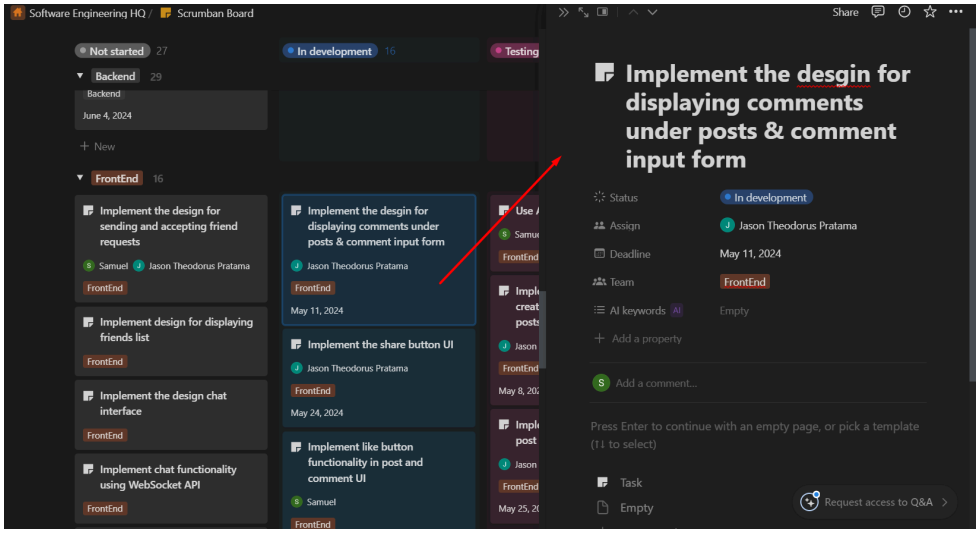
Task Card Template:



Screenshot of the example filled out task card:

Description:

- Task: Implement the design for displaying comments under posts & comment input form
- Status: In Development
- Assign: Jason Theodorus Pratama
- Deadline: May 11, 2024
- Team: FrontEnd



IV. Simulate a Scrumban Board:

Due to the content limit that can be shown in this report/file, we already published our simulation Scrumban Board of developing a new social media platform within 3 months using the Scrumban methodology which can be accessed through this link Scrumban Board Simulation.

V. Benefits or impact of generative AI on SDLC:

Generative AI has many impacts and benefits that have affected and revolutionized software developers' work. IT expertise has launched a new field that makes the most of generative AI and large language models, which is called *prompt engineering*. With prompt engineering, engineers can gather and gain generalized information quickly by just simply providing simple prompts. Based on research, data indicated that 73% of the respondents consider that AI and machine learning play an important role as crucial skill sets for their

organizations. Moreover, tools and platforms like ChatGPT, DALL-E, and GitHub Copilot derive many benefits that change how the developers work. In simple words, generative AI can accelerate the application creation process and allow coding specialists to focus on higher-level, creative, more complex activities.

The main 5 elements of SDLC in general are :

- **Planning and analysis of requirements**
  Software developers define the functionality and capabilities of a new application at this stage by taking into account user requirements and other relevant information.
- **Design**
  In this phase, the developers need to select the requirements of hardware systems in their application and the programming languages to facilitate the coding process.
- **Testing**
  During this phase, the main focus of the SDLC will be focused on testing the software functionally, detecting security issues, and finding ways to optimize the performance.
- **Deployment and implementation**
  Once it passes the testing process, the developers will bring a user's manual to the user and use it in a production environment. Moreover, developers will give some guidance for user training so that users will be used to their surroundings.
- **Maintenance and evaluation**
  During its initial deployment, there's rarely a case where this software will directly affect its perfect performance, so developers still need to monitor the application, collect user feedback, and bug-hunt to identify and fix any unforeseen functionality issues.

As it matures over time, generative AI brings many benefits to the SDLC that will become even more reliable and accurate, and these benefits include more than just code generation. The following are those benefits:

1. **It can help with various phases** like requirement analysis, design, development, testing, deployment, and maintenance.
2. **It assists in applying new concepts, new programming languages, and new packages.**

3. **Developer productivity can increase** since generative AI can guide lifecycle steps and assist with scaffolding code and code snippets.
4. **It helps alleviate some repetitive tasks** like testing and accumulating test data
5. **It provides alternative ideas** and helps with problem-solving
6. **It generates functional prototypes** for developers so they can more quickly test and iterate on ideas, thereby reducing an application's time to market.

As we know the above are the factors that indicate generative AI is excellent at many things today. However, SDLC must consider some challenges as they need some expertise, infrastructure, and procedural safeguards. Those challenges are:

1. **Legacy applications**

   Based on a weighted average, 9% of respondents stated that the largest difficulty in using generative AI is the inability to pivot legacy applications and systems.
2. **Data privacy**

   Companies need to be sure that the data and training input their generative AI system needs don't violate people's privacy before partnering with them or investing in other businesses to assist their generative AI ambitions.
3. **Ethical considerations**

   Generative AI has the potential for bias or the use of misleading information depending on its training process.
4. **Prompt injection**

   There will be some attackers that could inject prompts and control the software model unexpectedly if it runs without proper guardrails.

The existence of Generative Pre-Trained Transformers (GPT) can make huge changes to human lives. But no matter how great and effective it gets, we don't need to worry that it will replace our jobs in the future. Instead, the rise of this AI tells the world and the people that abusing AI such as GPT could put someone ahead of those who don't. For example, Large Language Models offer a lot of new tools for companies to use, emphasizing the importance of being able to stay ahead of others through the changes with the help of AI. But making these changes and improvements is not that easy, it is way more challenging, especially in changing

how the “people” act. These need to be done gradually, or else, it often fails because it’s hard to get people to change their habits.

In modern organizations, usually, companies try to do it in one of three ways:

1. **Configuration change:** This involves modifying a variable to change the result. For example, online stores could achieve this by incorporating their products from an external supplier into their listings.
2. **Process change:** This occurs when there’s a chance in a procedure executed by a bunch of people, often through a step introduction or elimination. For instance, adding an extra step in your sales activates if a salesperson’s discount exceeds 30% for each deal.
3. **Technology change:** This occurs when we write, add, or integrate a new software component. For instance, a bank could develop a specialized software that enables customers to apply for a mortgage and utilize their stocks as a payment within a single interface.

Technology change by far has been very difficult and strict for users. Among the various methods to drive change, adjusting configuration might usually be the simplest, of course, it needs to be followed by altering other processes, and then technological changes being the last yet the most challenging among all of them. This is because our software is either running in a live setting or it isn’t. Traditionally, code will be passed to the operations team for such testing and deployment that could cause delays and disconnection between each of the running processes.

Using AI tools like ChatGPT in building software could make things a lot faster and easier. ChatGPT acts like a helpful peer who’s always there and ready to support you with your coding. This means, developers will spend less time on boring neverending tasks and more on coming up with new creative ideas. For example, when developers need to connect to different services on their program (APIs), we can ask the co-pilot to write most of the code required to connect it, saving a bunch of time that may be spent reading instructions, handling errors, etc. This way, making software is quicker and less pressure. This approach not only could accelerate

development but also the entire cycle of the software creation since each of the processes is correlated.

Co-pilot, powered by OpenAI's Codex which builds on GPT-3, is designed for coding, and filling out code based on the user prompts. The developer team successfully crafted complex neural network codes for PyTorch, TensorFlow, and Scikit Learn, with the output being about 80% ready for execution. However, due to periodic retraining, the varying results from this tool pose challenges for consistent output. For simpler projects, avoiding the API documentation grind, ChatGPT could be considered flawless, providing ready-to-use code. It effortlessly generates code for integrating with Google Analytics API, by ruling out the need for manual documentation review.

ChatGPT is changing how we design and code, making things faster by providing any help or request from the users' prompts in no time. This AI-generated tool could save a bunch of time that is used to do research, making drafts, outlines, and executions. Besides doing tasks much quicker, it also helps professionals to focus on more challenging, tougher, and creative work. Even though the existence of GPT helps a lot, the need for human supervision is still needed, human skills also need to be there to make the program much better, and judgment isn't going away. Human skills are required especially for checking overall designs and making sure that everything is secure before proceeding to the next process. As technology keeps advancing, tools like ChatGPT will likely become a big advantage for those who can abuse it as early as possible, with integrity and responsibility using it.

The step-by-step methodological approach for the software development life cycles according to Rajbhoh, et al. (2024) is as follows; requirements-specific generation, design specifications generation, application code generation, and test case generation, in that order respectively. In this report, we will be outlining the prompt templates for all four process steps.

1. **Requirements specification generation**

   This step in the life cycle process takes in specification details and input requirements. Developers ought to specify various requirements or parameters of their program. Generally, the prompt template consists of the business domain

(such as wages, pension, insurance, etc.), requirements (scope of the program), and context (comprised of additional details such as geographical location) as inputs. An example of prompts to be generated can be as follows:

"Considering the application in <*Business Domain*>, context such as <*Context*>, and business requirements as follows: <*Business requirements*>, please generate the requirement specification."

2. **Design Specification Generation**

   This step in the life cycle focuses on generating design specification details based on input requirements specification. In general, it follows that there are 6 prompts to be generated in this process. The first 3 prompts focus on generating high-level descriptions along with detailed specifications. The following prompts will help generate input/ output data fields as well as screen flow details. The prompts, as suggested by Rajbhoj, et al. (2024) are as follows:

   a. My application domain is <*Business Domain*>. Consider the following requirement: Processes: <*Processes Spec*> and Rules: <*Rules Spec*>.
   b. Generate entities and relationship descriptions for the above requirements.
   c. Generate service descriptions for the above requirements.
   d. For each <*service*> in <*Prompt 2 Response*> generate further details such as input and output parameters.
   e. Generate screens for the above requirements.
   f. For each <*screen*> in <*Prompt 4 Response*> generate further details such as UI classes, attributes, buttons, and screen flows.

3. **Code Generation**

   Based on the 2 previous steps in the SDLC, the code generation prompt takes in the business domain, technology stack, entities, services, and screen details as inputs. The code can comprise UI code, service layer code, business logic, and so on depending on the developers' design specification generation. The code generated by the given prompt will be reviewed and integrated into the developers' IDE of choice to be modified as needed. Prompts usually depend on the developers' requirements, but a general template is as follows:

   "My application domain is <*Business Domain*>, and application architecture

specification is as follows: <*Technology Stack*>. Consider the following design specification: Entities: <*ER Spec*> and Services Specification: <*Services Spec*>. For each <*service*> in Services Spec, generate service class code for <*service*> and data access object class code for <*service*>."

4. **Test Cases Generation**

   In this step, we generate unit and system test cases based on input services and screen details generated from the previous SDLC processes. This step is rather straightforward, we generate various test cases along with test data, such as functional test cases, data validation test cases, and so on. After the test case generation for all services and screens are generated, the system test cases are then generated last. An example of a prompt for this step would be as follows:

   "My application domain is <*Business Domain*>. Consider the following design specification: Services Specification: <*Services Spec*> and Screens Specification: <*Screens Spec*>. For each <*service*> in Services Spec, generate test cases covering these test case types: <*test case types*>. For each <*screen*> in Screens Spec, generate test cases covering these test case types: <*test case types*>. Generate system test cases for the application."