Planning Game - RemindRX

Team Members: Jawad Kabir, Abrar Habib, Ishmam Fardin, Sibora Bobam, Ivan Chen

Today we are going to start our projects by writing User Stories and playing Planning Poker to estimate the size of those stories in Story Points. You can use either Firepoker or Planningpokeronline planning games.

(done) User Story Time

For your project you will need to break down the overall product vision into smaller units that are known as user stories. There is a simple template for writing user stories:

As a < type of user / user persona >,

I want < some goal >

so that < some reason / value >.

As a…	I want…	so that…
Regular user	to view all my prescriptions and their details	the directions and time to take are readily accessible
Regular user	to be able to add my prescriptions by scanning the bottle/prescription	the input fields can be auto populated including recommendations for when to take them
Regular user	to edit my prescriptions+frequency	I can stay up to date with my treatment plan
Regular user	to be reminded via sms when it's time to take my prescription	I don’t forget to take my medicine
Regular user	to receive notifications when my medication is running low	I can schedule a refill
Regular user	to be able to converse with a AI agent	I can get a simplified description of what my prescriptions are for

Regular user	to be able to mark prescriptions as taken	so that I can check if I already took a medication or not.

You will use this format to describe all user facing capabilities that your software will implement. Some teams also apply this approach when describing nonuser facing work that needs to be done for the project. My recommendation is to only use this format for user facing capabilities. Describe the technical work, setting up base node app, or configure Postgres, in concise, plain language focusing on technical constraints where necessary. When describing a bug, explain how to reproduce the bug, including any relevant stack traces or log info. If you have done the investigation and identified where in the code the bug is implemented include the pointer to the code in the bug report.

Story Points

We covered story points briefly in the lecture. They are an arbitrary point system that is unique for a specific project, team, and context. You want to agree as a team what 1 point of work is, it is all right to select something concrete for example, a web form which accepts input form the user, validates it, reports any errors, then calls a backend service to store the data, is a 1-point story. Then use this as a baseline for other stories. For example, if you estimate a bug fix in the validation logic for a web form that is less work than 1, call it ½. On the other hand, if you are implementing something like an application signup flow which includes multiple pages, followed by e-mail validation that is likely to be twice as much work so 2 story points, or even 4 story points.

- 3 pt - Scanning prescriptions from the camera (streamlit camera input) and removing background information (opencv)
- 4 pt- Developing/using a model that can extract just the text information from any given image
- 4 pt- Developing a model that can recommend best times to take a medication when given directions and prescription name/dosage
- 3 pt - Sending sms notifications using Twilio API/ SMTP (for now, a sample message. must be reusable)
- 3 pt - Using a CRON job (celery api) to be able to schedule send times for sms notifications
- 1 pt - Setting up a database schema for patients and their prescriptions
- 2 pt - Creating the database and hosting it (ensuring real time updating)
- 1 pt - Creating database credentials for each developer on the team to use
- 2 pt - creating the streamlit frontend for users to scan in their prescriptions
- 4 pt each page - Creating the frontend for which users can see their prescriptions, which includes designing, implementing the design, and connecting database information

- 3 pt - create the react component that will allow users to mark off the prescriptions they have already taken. Update database column for that prescription indicating it has been taken.
- 4 pt - Auth flow, allowing users to access their account on both streamlit and on dashboard. Would most likely need to confirm with the user to store information as well as get other necessary information.

Playing a game of Planning Poker

Once you have your user stories, and you have agreed on what a single-story point worth of work is, you are ready to play planning poker. For each user story you have prepared, do the following using one of the online applications mentioned at the top of the page.

1. The product manager presents the user story and answers any questions from the rest of the team.
2. Everyone except the product manager then assigns a story point size value to the story.
3. Reveal the result once everyone has voted.
    a. If everyone agrees, that is if everyone assigned the same number, great that is the estimate
    b. If there is a disagreement the person with the lowest and the highest value explains their reasons for giving the score, and the team votes again.
    c. If after two rounds there is not a single score you can choose your reconciliation strategy. Mean, Median, or Mode. It will not matter which one you choose, mostly because we are bad at estimating anyway.
4. Repeat for all user stories, or until you have run out of time.

Task	Story Points
Scanning prescriptions from the camera (streamlit camera input) and removing background information (opencv)	3
Automated SMS notifications when it's time to take prescription	3
Automated notifications when prescriptions are running low	3
AI agent that can communicate with user based on all associated prescriptions	4
Interface to view all prescriptions for a particular user	4
Interface to scan prescriptions and update a user's prescriptions	3
Recommend patients the best times to take their medications	4
Create a react component that allows users to mark prescriptions they've taken, and update the database	2
Auth user flow	4

Filling the Iteration or Release

Once you have estimated stories it is time to decide which stories to put in your release and how to organize in your next iteration. I am constraining the iteration and release schedule; the duration of the iterations will be two weeks from Monday to Monday, and the Release will be done every two iterations, that is every four weeks. During the semester you will have two releases of your software. The other constraint is how much time you spend building the product. This is a three-credit course, the expectation is that you spend at least as much time on the course outside of the classroom as you do in the class round, that is three hours a week, six hours per iteration, or twelve hours per Release. During your first iteration you will get a baseline for how many story points you can complete per week as a team, your team velocity. For the first iteration you can use a base line of two-story points per iteration per teammate. After the first iteration you can adjust this baseline based on actual performance.

To select the stories for the first release, identify a set of stories that together create a complete and compelling software product. I would recommend starting by building useful features first, the core of your product, and necessary features after, for example sign up flow. Then for the first iteration select user stories that must be done first so that other stories can be done later, for example if you have a web form to collect information and later a page which summarizes the collected information using statistical analysis then implement the web form first.

RemindRx: For the first iteration, the first four weeks of development, we aim to achieve a functioning MVP. We want to make sure users can scan their prescriptions, get the best recommendations for when to take their medications, have a realtime db update with this information, and send timely reminders to take their medications. We believe that these features allow the second iteration to flow smoothly and are ultimately the crux of the project.