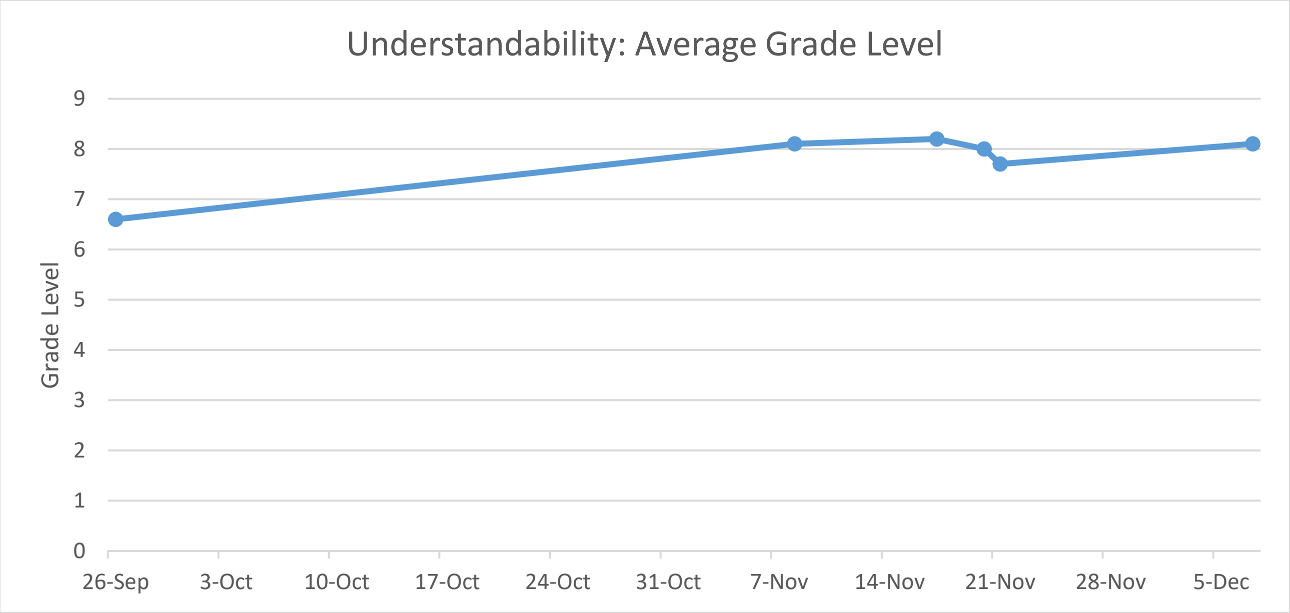
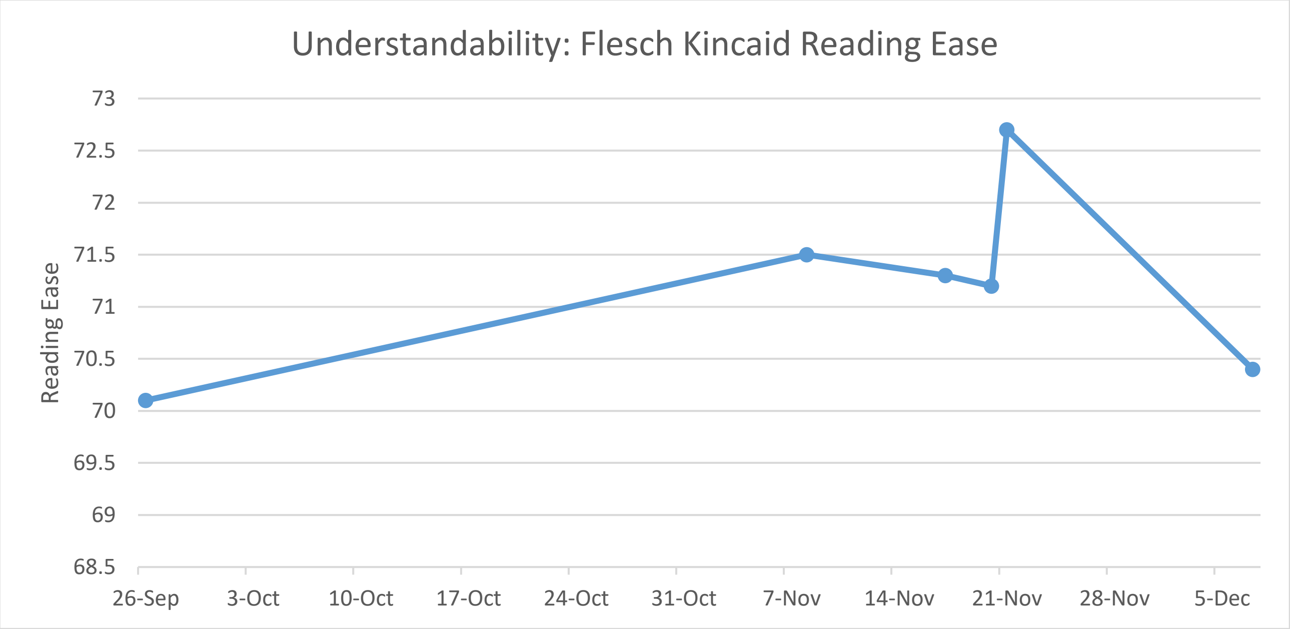
CSC506 Group 2 (Buckets)
Brandon Upp, Angus Dobson,
Nyalia Lui, Riddhi Shah, Yash Argawal

Metrics Charts & Graphs

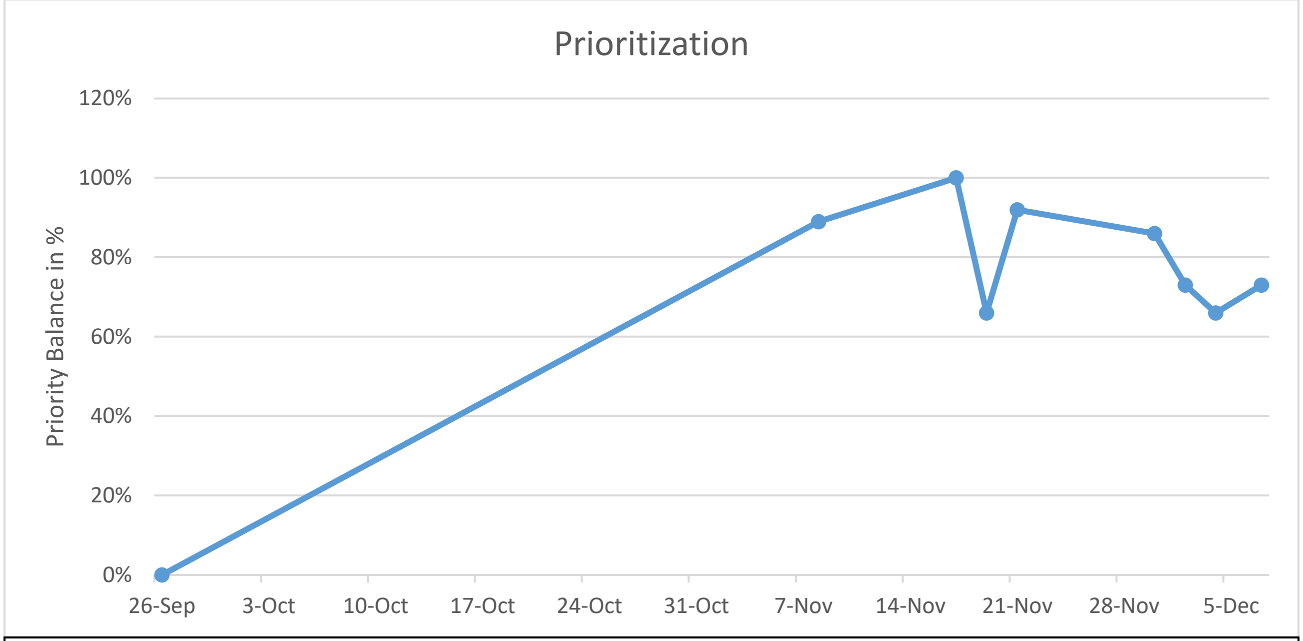
Requirements Metrics



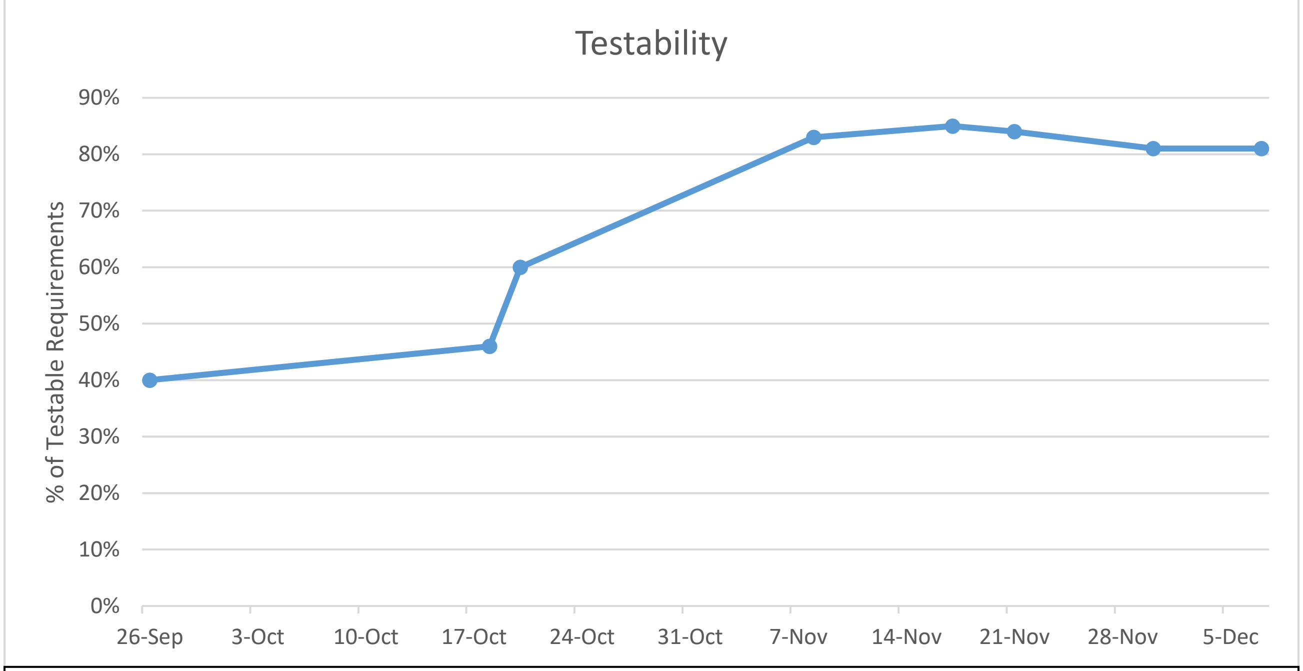


Understandability: We geared our language towards people with a minimum education of 9th grade. We're proud to say that our Average Grade Level barely exceeded 8th grade. Likewise, readers should have a comfortable experience scanning our requirements as the Flesh Kincaid Reading Ease was maintained in the 70's range which is akin to 8th-9th grade reading levels.

CSC506 Group 2 (Buckets)
Brandon Upp, Angus Dobson,
Nyalia Lui, Riddhi Shah, Yash Argawal



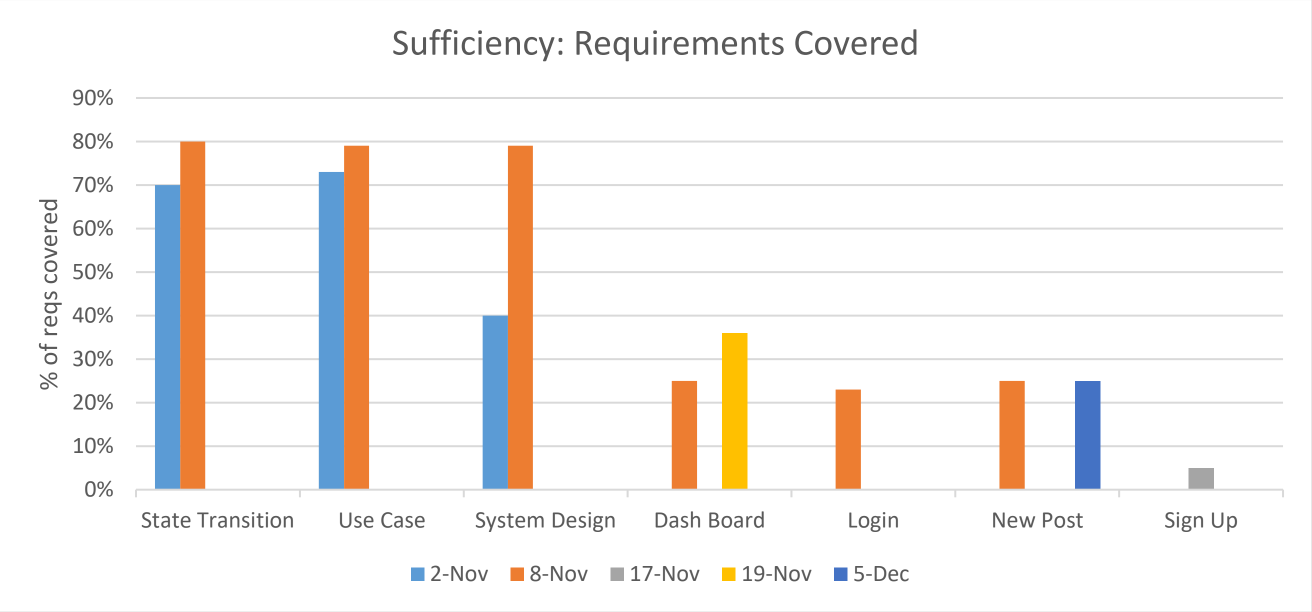
Prioritization: As our project began to unfold, prioritizing requirements became more important. Towards the end of the semester, our priorities began to drop as we have not closed our most recent iteration and this is where re-prioritizing occurs.



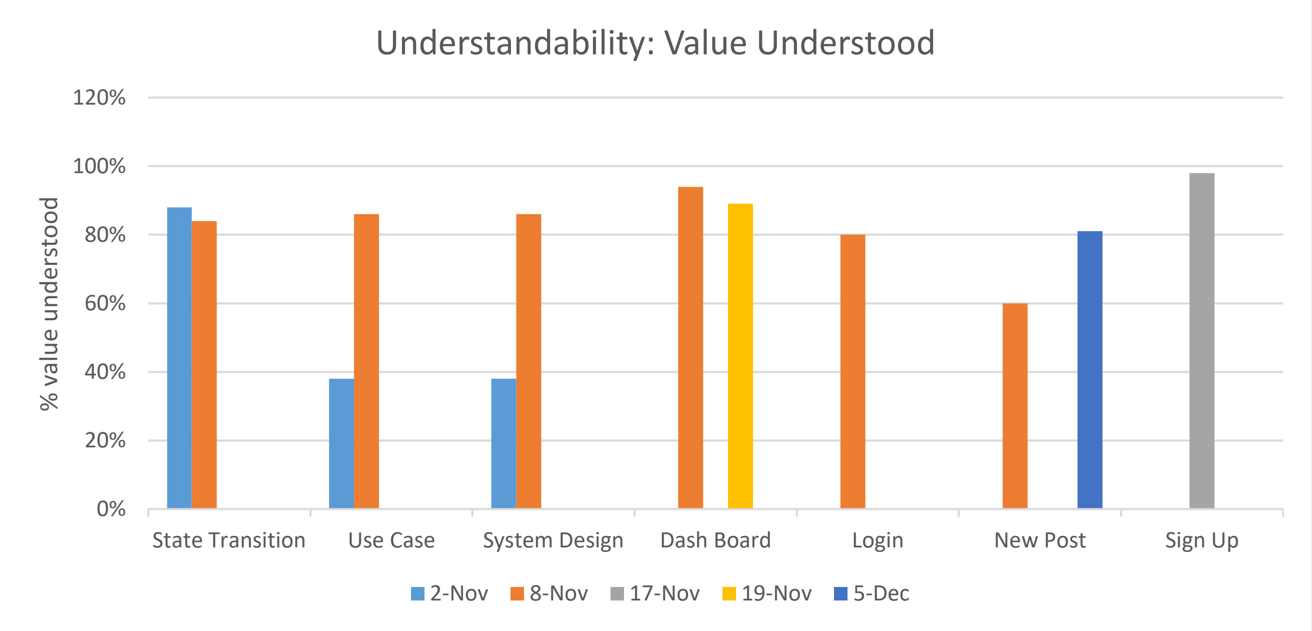
Testability: Here, our requirements increased in testability over time. The score hits a slight decline mid-November as we hit a big boom in implementing feature after feature where some of these had fewer testable requirements linked to them.

CSC506 Group 2 (Buckets)
Brandon Upp, Angus Dobson,
Nyalia Lui, Riddhi Shah, Yash Argawal

Design Metrics

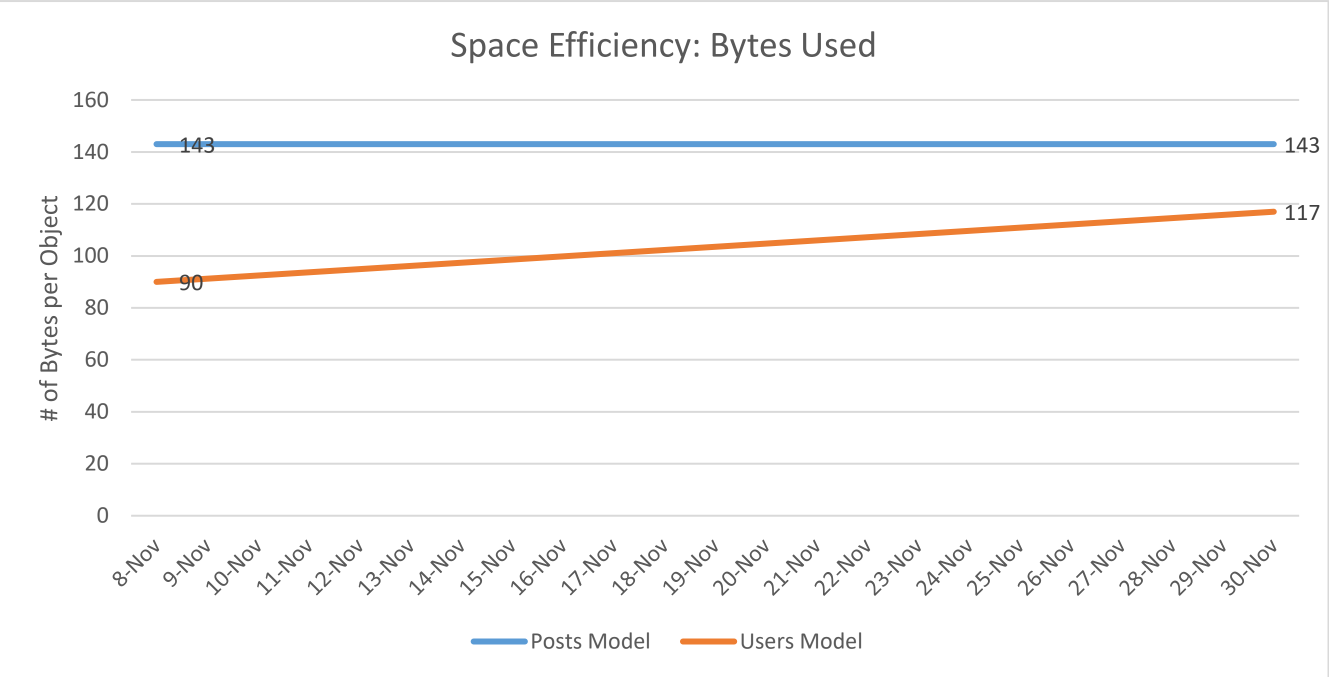


Sufficiency: Here we see that some design documents met more requirements than others. We think this happens because towards the beginning of the project, the team was evaluating the project on a larger scale so designs that affected the entire system were used (i.e. state transition). As more details were determined, the design documents did the same and were focused on specific features (i.e. Sign Up).



Understandability: This metric served as the primary motivator for modifying designs. If the design did not meet the 80% threshold then that design was remade. In special cases such as the Dash Board, changes were made to these as the application developed.

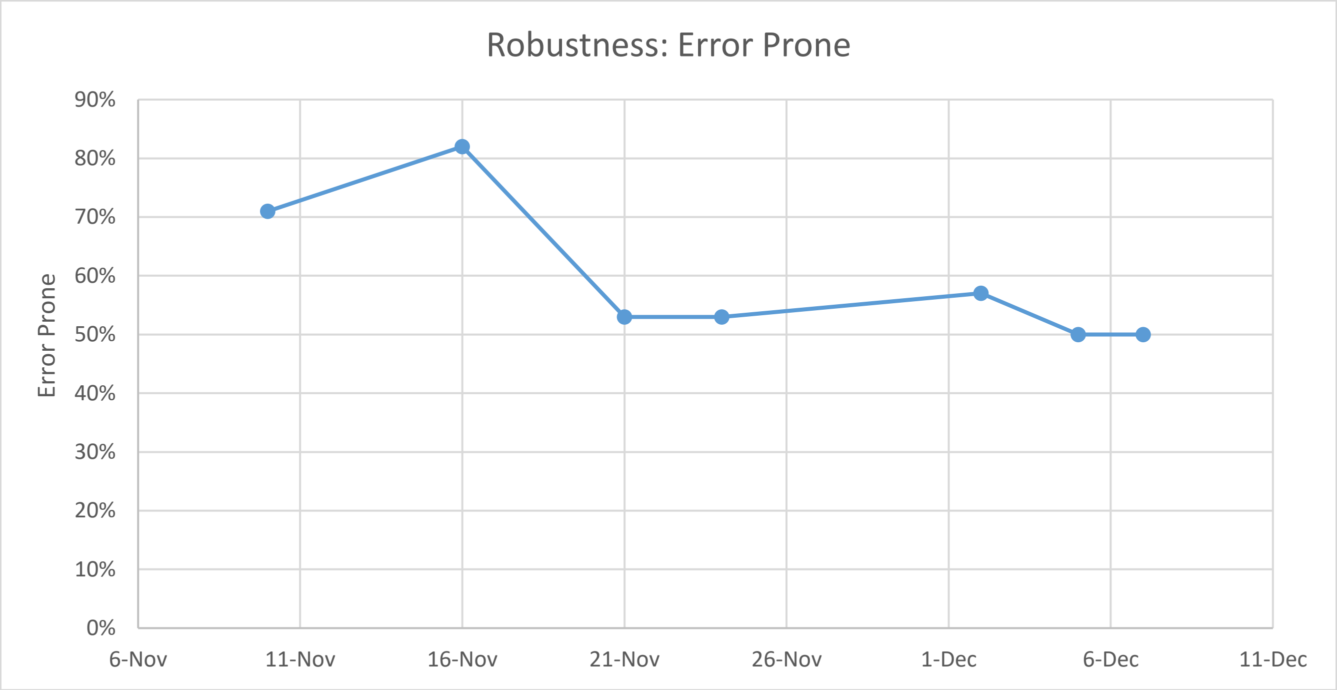
CSC506 Group 2 (Buckets)
Brandon Upp, Angus Dobson,
Nyalia Lui, Riddhi Shah, Yash Argawal



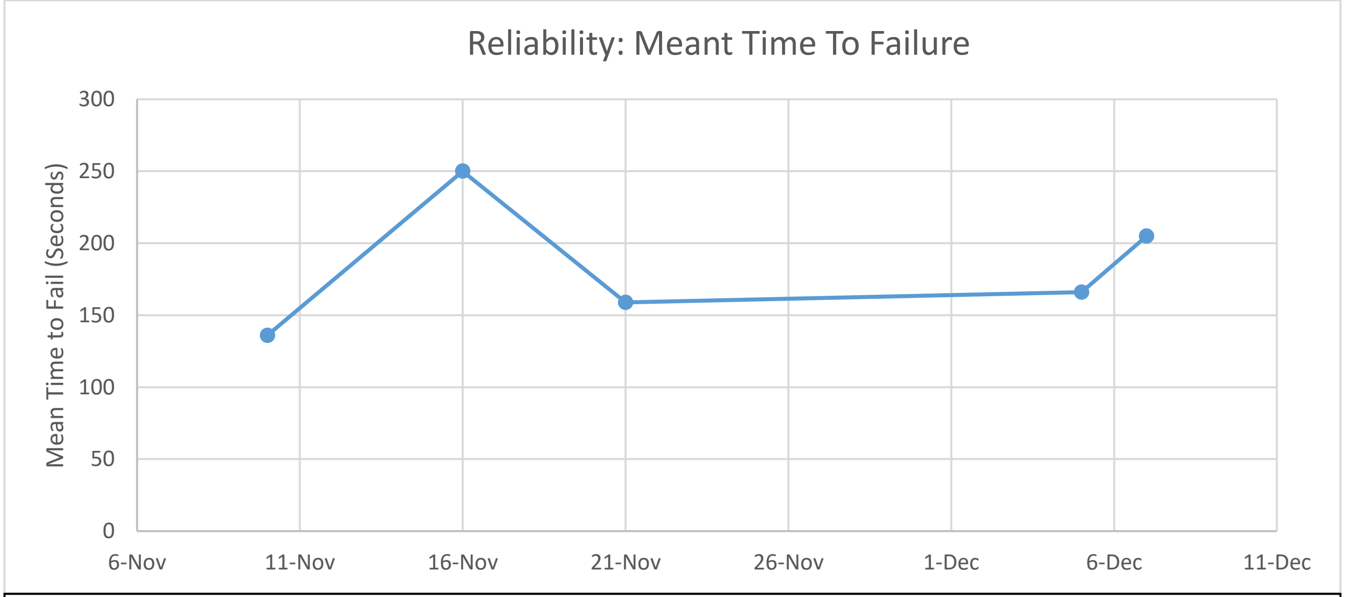
Space Efficiency: We measured this metric by the number of bytes each object takes on the database. For example, the number of bytes each User takes. Our Posts model did not change after its creation as it was made with the necessary detail to move forward. However, with our Users there is an increase in data needed for each User because Single Sign On options were implemented around mid-November and this requires more data in the database.

CSC506 Group 2 (Buckets)
Brandon Upp, Angus Dobson,
Nyalia Lui, Riddhi Shah, Yash Argawal

Implementation Metrics

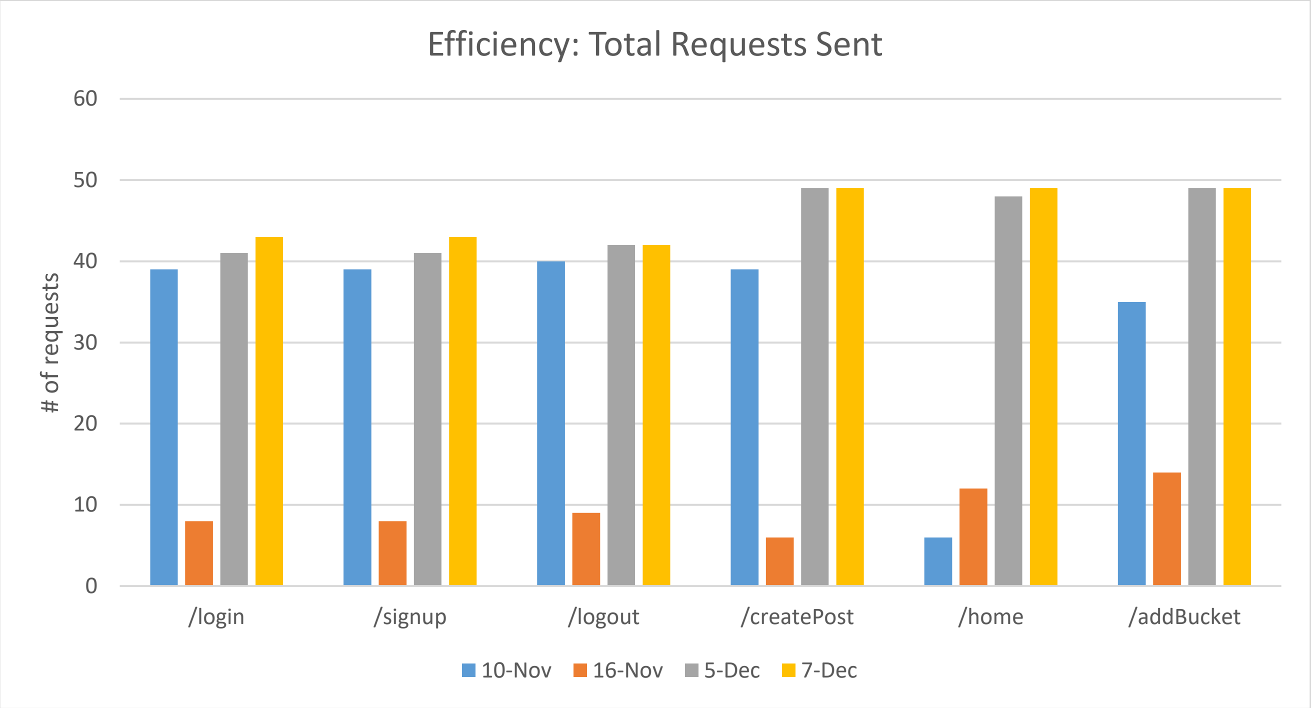


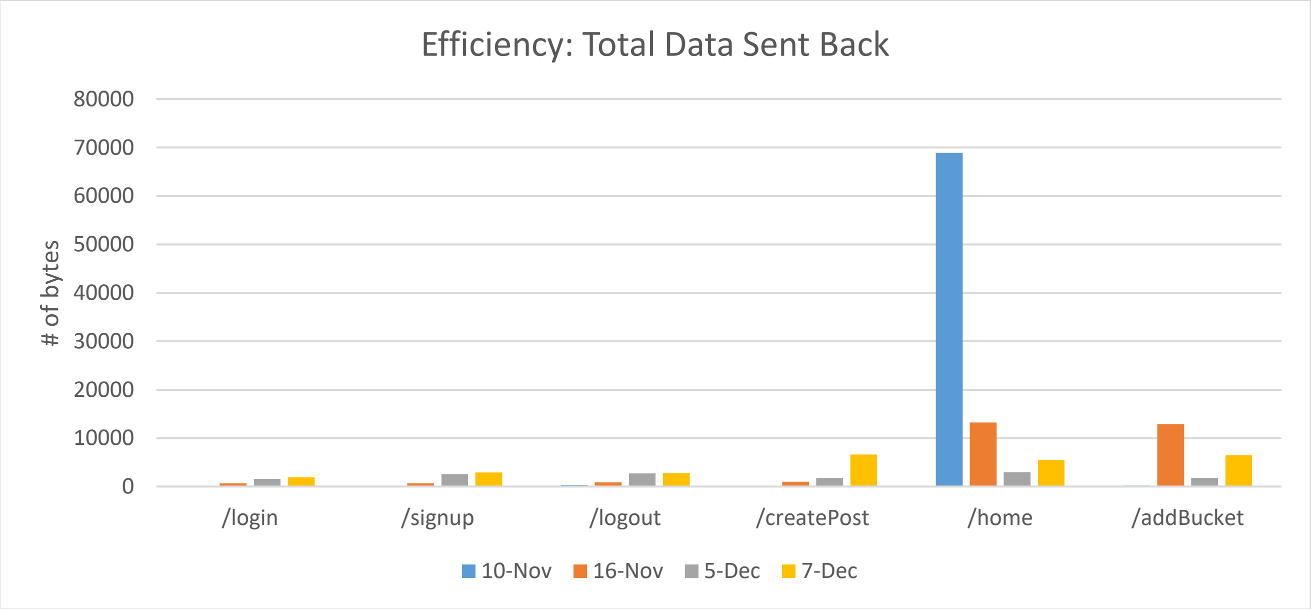
Robustness: We're proud to say that over time we minimized the likely hood that a user will experience an error on our application from its peak of 82% down to 50%. Moving forward, we believe that this can be lowered even further and ensure a pleasant experience for our users.



Reliability: Here we see that around the early stages of implementation, one could expect to have frequent failures with the application. More failures did occur towards the end of the semester as last minute features were quickly being implemented. However, we believe the user should not experience many issues with our app since there is a long run valley between mid-November and early December.

CSC506 Group 2 (Buckets)
Brandon Upp, Angus Dobson,
Nyalia Lui, Riddhi Shah, Yash Argawal

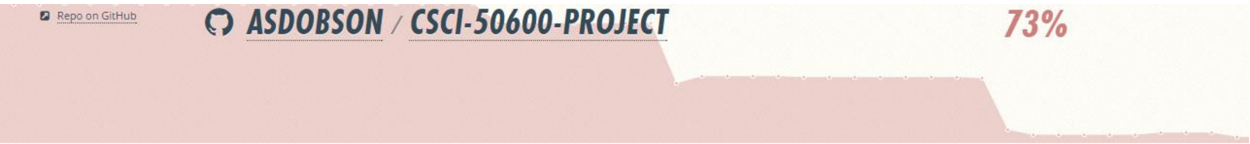




Efficiency: In the early stages of implementation, our application was sending a lot of data for a relatively small number of requests. Yet, drastic improvements were made such as on the /home page where most issues were. We're happy to say that currently the route that sends the most data is /createPost with 49 requests per 6600 Bytes of data. We believe this happens because after submitting a new post, extra data from the new post is added onto the /home page.

CSC506 Group 2 (Buckets)
Brandon Upp, Angus Dobson,
Nyalia Lui, Riddhi Shah, Yash Argawal

Code Coverage



Code Coverage: Up until December 3rd, we maintained a coverage above 90%. Following, coverage slowly fell due to last minute implementations. We believe we can easily get back to high code coverage in our next iteration.