Prof. Kaiser
Continuous Integration and Coverage - System Test Report
SYNC(Ruicong Xie-rx2119, Tianci Zhong-tz2278, Yue Cen-yc3171, Jiaxin Su-js4722)
December 10th

System Test Report

We did our system test by checking our application with all the user stories and non-functional requirements we have, and we received positive test results. Detailed test results are listed below:

I. User Stories

No.	Type	Description	Test Result
1	Inputs valid link(s)	User submits valid link(s) by the given text field in the index page; the link(s) are verified as valid by TinyUrl .	Perform corresponding functionality, passed
2	Inputs invalid link(s)	User submits invalid link(s) by the given text field in the index page, and TinyUrl displays error message(s).	Perform corresponding functionality, passed
3	Receives shorten link(s)	After the inputted long link is verified valid, TinyUrl jumps to the specific page for that converted link and displays converted link(s) to user	Perform corresponding functionality, passed
4	Shares shorten link(s) on the main feed page	User clicks on share button on the specific post on the main feed page, and that post get redirected to other social media platforms	Perform corresponding functionality, passed
5	Opens main feed page	User opens the main feed page; all the converted link(s) display as a list on the console	Perform corresponding functionality, passed
6	Track activity history for specific link(s)	User clicks on specific link(s) on the Private Url Block; the console switches to a new page and display detailed information for the link(s)	Perform corresponding functionality, passed
7	Delete specific link(s)	User deletes specific link(s) from the Private Url Block; the Private Url Block erases the link(s)	Perform corresponding functionality, passed
8	Post shorten link(s)	User posts specific link(s); corresponding post about the link(s) shows up in the main feed page, allowing other users to comment, like	Perform corresponding functionality, passed

9	Set link(s) private	User sets link(s) to be private; corresponding post about the link(s) will not show up in the main feed page	Perform corresponding functionality, passed
10	Set link(s) public	User sets link(s) to be public; corresponding post about the link(s) shows up in the main feed page	Perform corresponding functionality, passed
11	Delete post(s)	User deletes post(s) about his/her link(s)	Perform corresponding functionality, passed
12	Like post(s)	User likes shared post(s) on the main feed page	Perform corresponding functionality, passed
13	Unlike post(s)	User unlikes shared post(s) on the main feed page	Perform corresponding functionality, passed
14	Comment on post(s)	User comments on shared post(s) on the main feed page; comment will be saved at the bottom of the post(s)	Perform corresponding functionality, passed
15	Delete comment on post(s)	User deletes comment on shared post(s) on the main feed page; corresponding comment will be deleted from the bottom of the post(s)	Perform corresponding functionality, passed
16	Open leadership board	User opens leadership board from the index page or the main feed page; TinyUrl displays top 10 shared links by the click activity history in the system	Perform corresponding functionality, passed
17	Log In	User logs into the TinyUrl by their email address and password	Perform corresponding functionality, passed
18	Log Out	User logs out from his or her TinyUrl account any time he or she wants	Perform corresponding functionality, passed
19	Shares shorten link(s) from specific link page	User clicks on share button on the page for that specific link, and converted link get redirected to other social media platforms	Perform corresponding functionality, passed

II. Non-functional Requirements

We use MongoDB and Redis for our non-functional requirements. Based on the CAP Theorem, our MongoDB supports the consistency and partition tolerance of our system. Compared to application using SQL database, our system is highly scalable. On the other hand, we cached our IO request in Redis, which is another NoSQL database we used in our system. It is a in-memory key-value NoSQL database with performance comparable to volatile data storage that helps boost the performance of our application greatly. Both of MongoDB and Redis functions smoothly during the system test, supporting corresponding functionalities in our application.