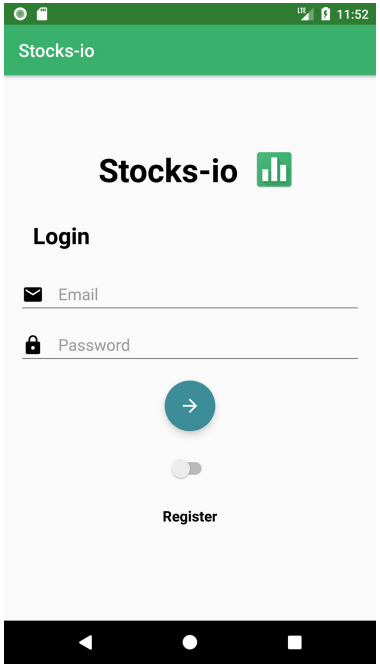
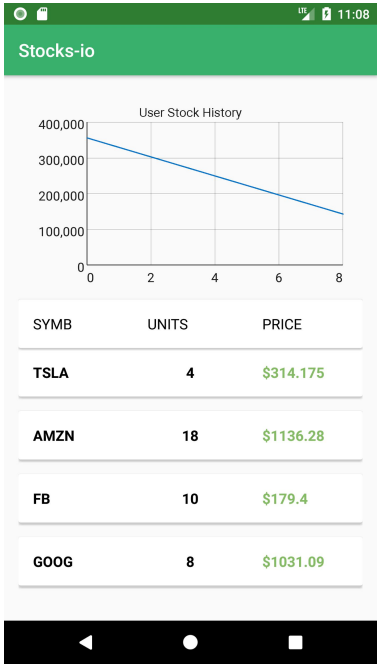
Stocks.io

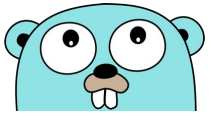
Dave Machado, Darrien Glasser

About

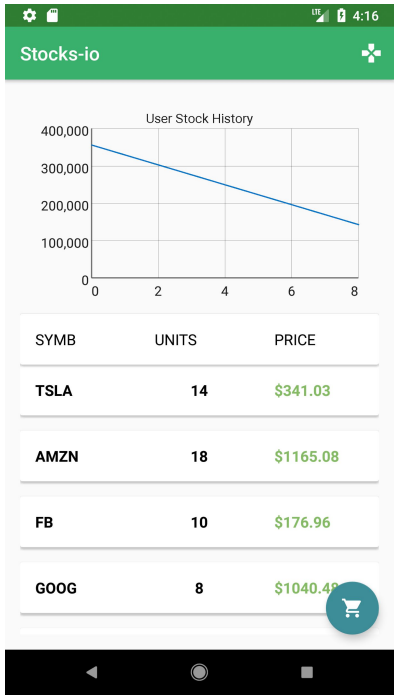
- Stock Market Simulator
  - But without the need for real money!
- Compete globally
  - Real time leaderboard with total assets

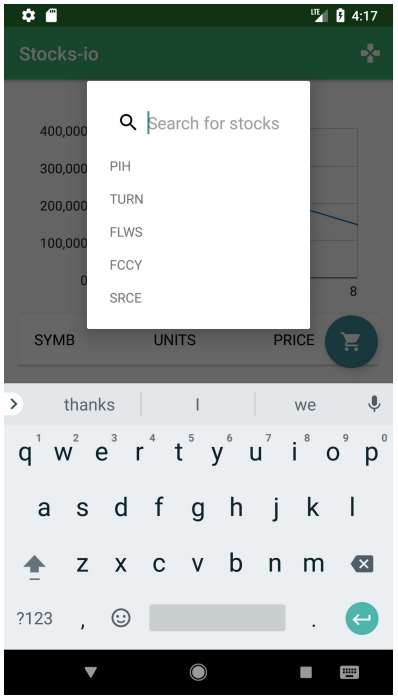


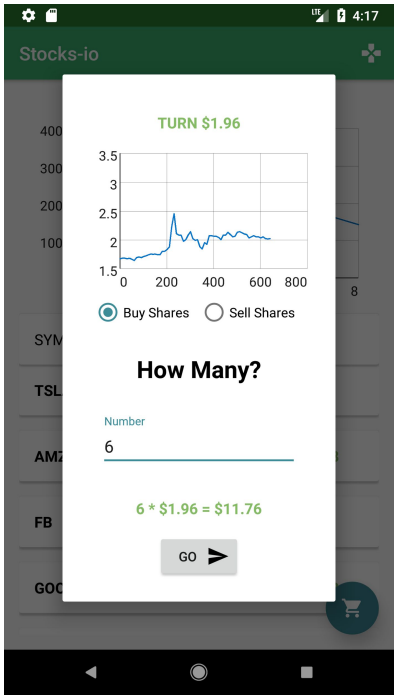


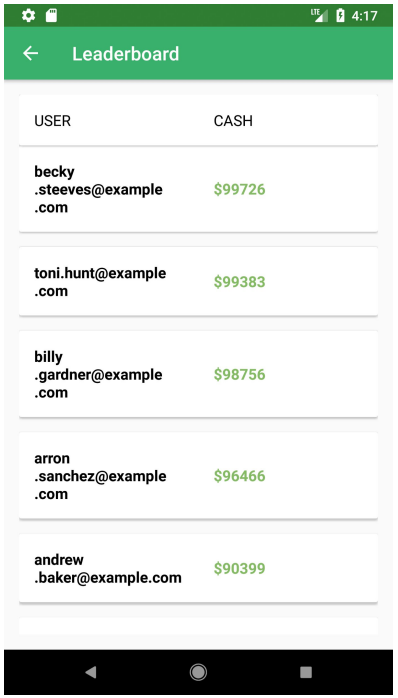


Buy and Sell Stocks - the full experience!









Compete with others on a global leaderboard!

Robust Testing

- Wanted to try and simulate industry-standard testing paradigms
- Implemented Integration, Unit, and Production testing schemes
- Integration:
  - Travis CI - once every day, all source code is pulled and run against a bash script of assertions
- Unit:
  - Server Unit Tests - Golang server has built-in `*_test.go` tooling to run small unit tests against different functions defined on the back-end.
- Production:
  - Generated fake user data via REST API to verify application could withstand high levels of user traffic and database access.

Challenges

- Goroutines are executed at different times via the operating system's built-in core scheduler.
  - Multiple producer, multiple consumer endpoints could block against one another if "scheduling" did not occur in the correct order
  - Solution: use Gochannels as a pipe to indicate to the main thread that any and all gorountines have finished using a particular resource/database access.
- JVM language interop was difficult to pick up at first
  - App written in both Kotlin and Java
  - Solution: migrated troublesome parts to Kotlin to preserve native language support
- App required synchronization of multiple asynchronous calls
  - Required careful thread moderation, with explicit fail-safe code to prevent app crashes

Things We Learned

- Go has a lot of great native support
  - While it offers concurrency out of the box, using it comes with great responsibility.
  - If not implemented carefully, goroutines can spiral out of control and either lend the process to deadlock or core dump.
- Multithreading endpoints improves server performance significantly!
  - Since we don’t have to wait for one request to finish before starting another, independent users can access various services at the same time with minimal latency.
- Using reverse proxies to obfuscate common ports increases security
  - Instead of accessing MariaDB over the known 3306 port, we can hide the real port being observed and reroute any negative traffic to a null-safe endpoint
- Kotlin null safety is extremely convenient
  - Null checks are built into the compiler. As a developer I never have to worry about whether or not a variable will be null.

Thank you for a great semester!