Benjamin Dewey

The application is a virtual mailbox service network. Clients can request their own personal mailbox from the service, search the list of existing mailboxes, and send and receive mail. The CRUD operations are as follows. Create: a client can ask the service to create a mailbox with a unique mailbox name supplied by the client, at which point they will receive their mailbox password; Read: a client can read the contents of their mailbox provided they have the mailbox name and password, and they can view/search the list of mailboxes that currently exist; Update: a client can send mail to another mailbox provided they have the recipient mailbox name and they have their own mailbox in which to stage the mail before delivery; Delete: a client can destroy their mailbox so that it no longer exists.

When a client wishes to send mail, any delivered mail currently sitting in their mailbox is first returned to them, and then the mail is put into their mailbox. The service runs a job every two minutes that visits all mailboxes and transfers any outgoing mail into the correct destination mailbox. When a client sends mail the service responds with how much time is remaining until the next delivery job will run. A unit of mail consists of user and auto generated values. They are a timestamp of when the mail was created, the sender's mailbox name, the name of the mailbox to send the mail to, and a message written by the sender.

The client and service communicate with each other via RPC, and the RPC system itself is implemented using the gRPC framework which generates Python client and server stubs from service definitions written in the Protocol Buffer interface description language. Protocol Buffers are a data serialization system, like JSON or XML. Client interactions with the service are handled by a command line utility. The RPC calls are RegisterMailbox, RemoveMailbox,

GetMail, SendMail, and ListMailboxes. These calls correspond to the available commands a user can pass to the mailman.sh command line utility: add, rm, get, send, and ls.

The service is deployed on Google Cloud Platform and is listening for any and all clients to connect to it. There is no restriction on who can access the service, however mailbox access is password protected. Anybody can register a mailbox or view/search the current list of existent mailboxes, but getting mail, sending mail, and removing a mailbox requires that the client send along their mailbox password. The client program ensures that command line arguments, which ultimately make their way to the service as fields in the Protocol Buffer, are all present. It is up to the service to make sure that the fields are functionally valid. For example, if a user tries to register a mailbox that already exists, supplies an incorrect password, or tries to send mail to a mailbox that does not exist, the service response will include an error message for the client to show to the user.

All of the data that represents the state of the mailbox network is stored in memory, so if the service goes down all data will be lost. The service is also vulnerable to concurrency issues because some operations that are conceptually atomic are not so in practice. For example, if two RPC calls come in at the same time, one to get mail from mailbox X and one to remove mailbox X, then if one thread removes mailbox X while the other is still reading mailbox X the service would likely throw an exception and crash.