
# Salesforce Mock API

**Trevor Kerr, Joseph O'Brien, Miles Bishop**

**June 18, 2019**

## Table of Contents

# Introduction

## Client

Salesforce is the company behind the world's #1 customer relationship management platform. Their software is cloud-based, making it easy for teams to setup and use. Salesforce offers a wide variety of products to help improve productivity, communication, and organization.

Salesforce is one of many companies to make the move towards a microservice architecture. Loosely coupled services means easier development, however, testing and integrating these services is not so easy. In order to test a new microservice, one must also set up the other microservices that it will be communicating with. This can take a long time and presents a significant challenge for developers trying to maintain microservices.

## Product Vision

Our product aims to expedite the process for testing microservices by creating an API that features runtime configurability for mocking other services and gRPC Remote Procedure Call (gRPC) support over HTTP/2. However, our client also specified a feature wishlist which gave us these additional features to implement:

- Automated configuration through Open API Specs (Contract Loading)
- Browser based UI for summarizing endpoints
- Simulating connection timeouts
- Time constrained testing
- Persistent datastore for loaded endpoints
- Triggers (Actions to be performed when certains request is made)
- Value Generation (Generate value for a specified data type)

Our client should be able to create multiple instances of our product and mock any services needed to create the appropriate testing environment. The client should also be able to get meaningful information back from the product regarding the results of API activity.

## Functional

The Functional Requirements are as follows:

- Be able to be configured to accept certain requests
- Be able to respond with specific responses
- Inform the tester of the test results

- Contract testing - given an API contract, at the end of tests, verify that all requests and responses match that contract
- Support gRPC requests

**Non-Functional**

The finished product comes with documentation so that it can be easily set up and used. In addition, there are a few backend details that were observed. The product needed to be written in Java using the spring framework, and was as RESTful as possible. Some other non-functional requirements follow:

- Documentation on how to use Mock API
- List of API endpoints
- Target a RESTful API pattern
- Written in Java 8
- Utilize the spring framework

## System Architecture

**Description**

The design of our Mock API service can be shown using two flow charts, see Figure 1 below and Appendix A. The first flowchart shows how our service fits into the client's current service. There are two options for the use of our system; the first option is that their software will generate a request which will be sent to our Mock API, then our API will send a response back. The second option is for the client to access the API manually to add endpoints, remove endpoints, or check the test results. The API will perform the appropriate task and then show the user the results of that task. The second flowchart shows how the request is handled. If the API receives a ‘new endpoint‘ call then the service will parse the HTTP/JSON file and create a new endpoint for it. It will then send a response of ‘Successfully created’ to the console. If the API receives a call to ‘delete’ an endpoint then the service will parse the HTTP/JSON file and remove the correct endpoint. It will also print a response of ‘Successfully Removed’ or ‘invalid id’ if the incorrect id was given. If the API receives a ‘test API’ call then the service will check if the endpoint exists, and if it does, it responds with data given when setting up the endpoint. The results for the API test call will be stored in test results. If the API service receives a ‘results’ call then the data from all the API test calls is compiled and sent to the user as either a JSON file or through a visual interface.

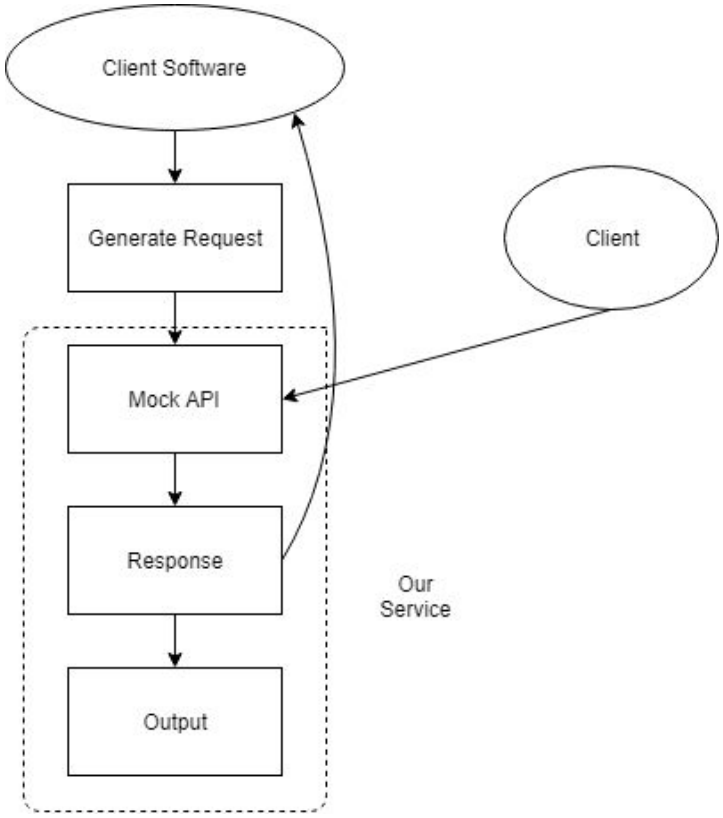



Figure 1: Our Service and the Client's Service

## Technical Design

### Testing Endpoints

Handling endpoints created by the user is one of the most important parts of this project. This process is shown in detail in Figure 2 below.

In order to accomplish this, we needed an endpoint capable of catching all the user-created endpoints. Any request not captured in our predefined endpoints has their path and method checked to see if it matches any user-created endpoints. If the endpoint is found, we need to check if it is a gRPC request. If it isn't, then we check if it contains any path variables. This is done by using a regex to check for any "{" or "}." If these are found, the variables from the request are stored so that the user can check what data is being used for this endpoint. We then set the response headers and body, and send that data back. When this is finished, the time when the endpoint was hit is stored and the number of times it has been hit is incremented.


Figure 2: Handling a Request

**gRPC Endpoints**

gRPC is a framework that speeds up data serialization for transport over a network. It can be used as a substitute for JSON or XML. In order to support gRPC requests, we needed to add HTTP/2 support and the ability to compile proto files. These files define the gRPC service that we need to emulate, and can be compiled to Java

classes. These files are then added to our project at build time so that they can be used to serialize the data to send back. This process is shown in Figure 3 below.

We decided to use the same request mapping for all stored endpoints, so gRPC (HTTP/2) and HTTP/1 requests go to the same place. In order to filter gRPC requests, we used a flag that users must raise when creating a gRPC endpoint. The user must also supply the name of the class that the protobuf compiler will create at runtime and use to serialize the response. Using Java reflection, our server instantiates a builder for that response object. Then, we take the desired response and convert it to JSON. This JSON is passed to a function that merges the JSON with our builder object. This object is then built into a protobuf response object and returned over HTTP/2.

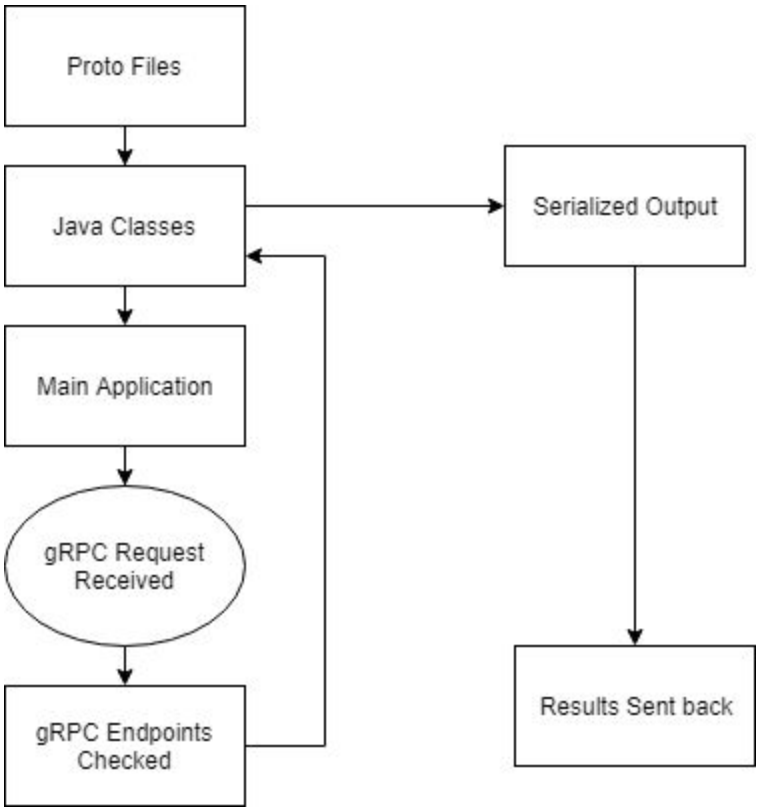



Figure 3: How gRPC Request is Handled

# Quality Assurance

### Goals

The quality assurance of our product follows these main goals:

- Summarize the strategy
- Provide information on testing tools used
- Summarize testing risks

### Strategy

Our quality assurance strategy mainly consisted of unit level tests in order to verify the functionality of individual components. The summary for each of these tests can be seen in appendix B. As we added new features, we performed integration and progressive regression testing in order to verify that other functionality was maintained. We also employed functional tests in order to assure that the product meets business requirements.

### Risks

Since attempting to deliver an easily configurable mock microservice, there are some risks associated with the dependability of the product. If the product fails to mimic certain microservice behavior, then Salesforce developers might find it more efficient to write their own mock services. This could be especially true if our documentation is not helpful for those that may want to extend our product to meet their specific needs. We could have also failed to test the product thoroughly enough to exploit potential inefficiencies or bugs.

# Results

There were some features that we were not able to implement due to time constraints. These features, such as triggers, value generators, and other additions to endpoints were not critical features according to the client.

### Features that we did not have time to implement

- Triggers
    - The ability to change the responses of certain endpoints when others are hit. This could be implemented in the future with some changes to the Endpoint class.

- Value Generators
    - Given something like ${email} as a response type, the service would generate a random email. This would be harder to implement, but it could be done.

**Performance testing results**

Our QA plan worked well, we were able to get cover 96% of all lines in the code base with tests. All of the JUnit and functional tests are passing.

**Results of usability tests**

The client is happy with the product, although there is still work that could be done to improve it. The service is working well, and all of the functional tests are passing. For future work, the client could add the features that we did not have time to implement. There were several features that the client wants that were outside of the scope of this project such as dynamic port binding and a full web UI. These could be implemented, however, they would take some more work due to the many complex parts required for each.

**Lessons Learned**

Spring is a very robust and easy to use framework. Once the framework is set up, all of the features that we needed were easy to add. The framework does a lot of the work for you, allowing you to quickly setup your project. Without spring, we would not have been able to implement as much as we did.

Java is a very powerful language, but with lots of boilerplate. It let us do everything that we needed to do, but could be very verbose. Also, there is a plethora of Java libraries, everything that seemed complicated for us to do manually had a library to help us. During the project, we had a problem with time and had to switch libraries. If we had looked at all of the options for a solution before we choose one we would have avoided rewriting code.

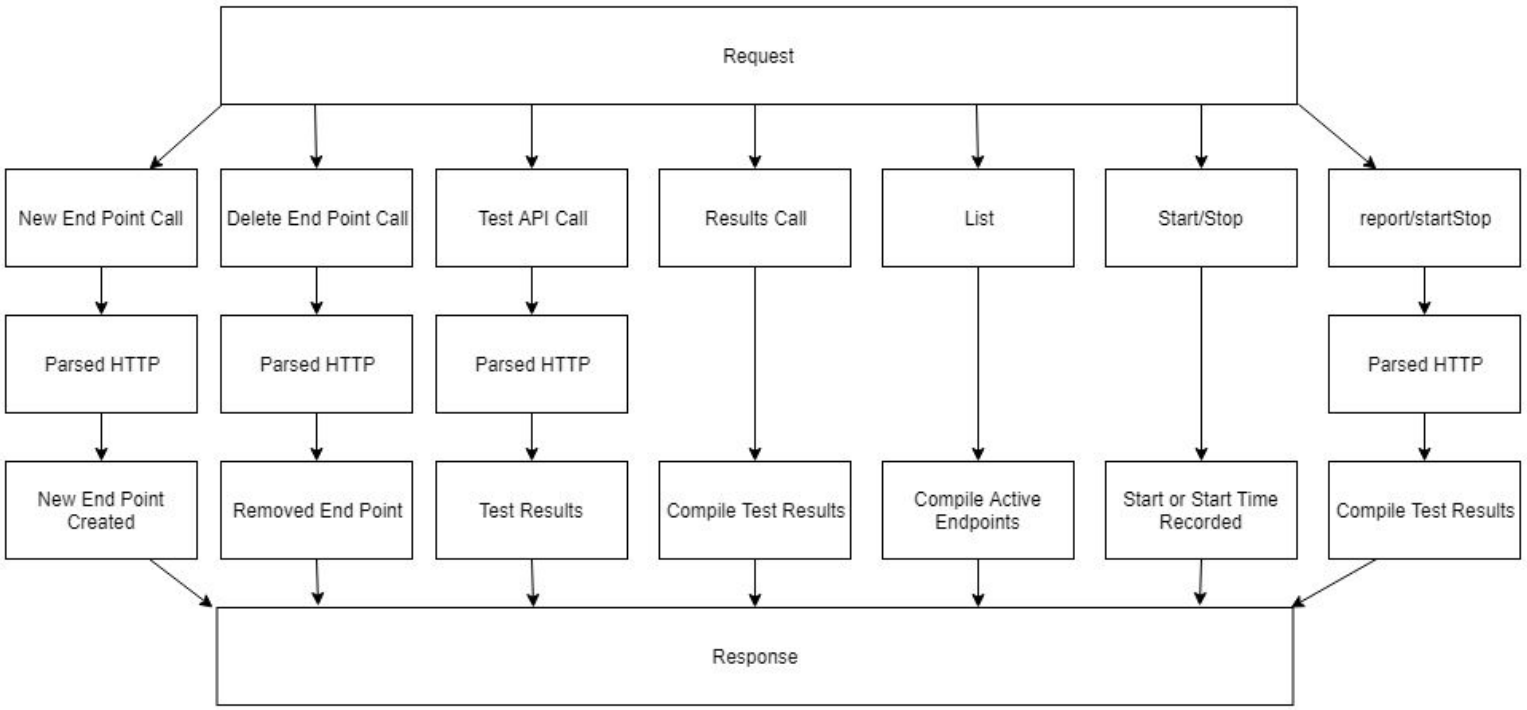
gRPC support was perhaps the most difficult task. We had to find a way to configure our Spring Boot application to accept HTTP/2 requests and then return the correct serialized response using protocol buffers. Serializing the responses was tricky. We used Java reflection in order to create Java objects from the proto files supplied. Then merged the desired response data with the created object. Most gRPC applications do not include this step because developers know what services will be implemented. It took us about a week to get this working. This was one of the critical features, and we waited till week 5 to start work on it. Had we started to develop this

feature earlier, we would've had more time to familiarize ourselves with the challenge we faced and write the appropriate test cases. This taught us that it is not a good idea to put aside critical feature implementation, especially when the feature relies on a framework that is foreign to the entire team and relatively new.

We learned that Agile is an effective way to manage a team. However, we also learned that is important to maintain a routine that keeps the team in sync and focused on the right tasks. There were days where we forgot to do a formal stand up. We also forgot to do some weekly retrospectives.

We also learned the importance of effective client communication. Our best resource was our client. There were times where we needed to be more effective communicators in order to keep the client and ourselves on the same page.

# Appendix A:Our Service



# Appendix B: Test Items

- **Endpoints (Spring @RequestMapping within a @RestController)**
  - *Creating Endpoint (/when POST)*
    - **Pass Criteria**: endpoints are created with the appropriate data and stored
    - **Deliverables**: jUnit test using MockMvc
  - *Deleting Endpoint*
    - **Pass Criteria**: endpoint for provided id is set to inactive
    - Deliverables: jUnit using MockMvc
  - *Report of Server Activity*
    - **Pass Criteria**: returns json of all active endpoints and their data as well as unexpected requests
    - **Deliverables**: jUnit using MockMvc that checks for 200 status code
  - *Report Web Page*
    - **Pass Criteria**:
      - request yields a well-designed web page that contains correct data for active endpoints
      - Also need to verify that user can delete endpoints using links/buttons
    - **Deliverables**: jUnits tests using MockMvc (not sure about UI tests yet)
  - *Report Using Time Constraints*
    - **Pass Criteria**: returns json of all active endpoints and their data as well as unexpected requests that occur between a given timeframe
    - **Deliverables**: jUnit using MockMvc
  - *Listing Active Endpoints*
    - **Pass Criteria**: returns correctly formatted json containing all active endpoints with correct data
    - **Deliverables**: jUnit test(s) using MockMvc

- *Making Requests to a Created Endpoint*
    - **Pass Criteria**: response contains the expected behaviour and any triggers, path variables, or request parameters are handled appropriately
    - **Deliverables**: jUnit tests using MockMvc and MockRestServiceServer

- *Verify API Contract Test Results*
    - **Pass Criteria**: contract are properly verified and reported
    - **Deliverables**: jUnit test using MockMvc (and maybe MockRestServiceServer)

- **Intermediate Items (Methods used within @RequestMappings)**
    - *Making Endpoint INACTIVE*
        - **Pass Criteria**: State of endpoint corresponding to passed in ID has been successfully changed to INACTIVE
        - **Deliverables**: jUnit test case(s)

    - *Adding Endpoint*
        - **Pass Criteria**: State of new endpoint is ACTIVE and it exists in endpoints map
        - **Deliverables**: jUnit test case(s)

    - *Adding Hit*
        - **Pass Criteria**: call increments request counter for endpoint and creates and adds timestamp for hit
        - **Deliverables**: jUnit test case(s)

    - *Adding an Unexpected Endpoint*
        - **Pass Criteria**: unexpected endpoint is created or modified with timestamp of request, request counter is incremented, and the state is changed to UNEXPECTED
        - **Deliverables**: jUnit test case(s)

    - *Getting Map Containing All Endpoints*
        - **Pass Criteria**: return endpoints map
        - **Deliverables**: jUnit test case

- *Getting List of Active Endpoints*
    - **Pass Criteria**: returns list of all active endpoints
    - **Deliverables**: jUnit test case

- *Filtering by Time*
    - **Pass Criteria**: returns properly list of endpoints that fall within the passed in start and stop times
    - **Deliverables**: jUnit test case(s)

- *Checking if Endpoint Meets Time Constraints*
    - **Pass Criteria**: adds filtered times if appropriate and returns copy of endpoint with filtered timestamps
    - **Deliverables**: jUnit test cases

- *Get Requested Endpoint if it Exists*
    - **Pass Criteria**: returns the correct endpoint object if the request matches an existing ACTIVE endpoint
    - **Deliverables**: jUnit test case using MockMvc

- *Getting Stack Trace of Requests*
    - **Pass Criteria**: return list of endpoints that have been called
    - **Deliverables**: jUnit test case

- *Get Endpoint for ID*
    - **Pass Criteria**: returns the endpoint corresponding to the passed in ID
    - **Deliverables**: jUnit test case

- *Getting String of Requests Stack Trace*
    - **Pass Criteria**: returns json formatted stack trace (list of called endpoints)
    - **Deliverables**: jUnit test case

- *Checking if Endpoint Already Exists and is ACTIVE*
    - **Pass Criteria**: returns the endpoint object for passed in ID if endpoint exists and is ACTIVE, null otherwise
    - **Deliverables**: jUnit test cases

- **Other Functionionality**
  - *Connection Timeouts*
    - **Pass Criteria**: system waits the requested time duration and returns the correct response
    - **Deliverables**: jUnit test cases

  - *gRPC Support*
    - **Pass Criteria**: product is able to accept and return gRPC requests (HTTP 2)
    - **Deliverables**: jUnit test file/case(s)

  - *Value* Generators
    - **Pass Criteria**: values are generated and stored appropriately
    - **Deliverables**: jUnit test cases using MockMvc

  - *Contract Loading:*
    - **Pass Criteria**: a swagger spec can successfully be loaded resulting in a configured mock server that works as specified
    - **Deliverables**: jUnit test case with the use of MockMvc

# Appendix 3: Report Feedback

Team Reviewed: Salesforce 2

Reviewed by: Alejandro Caraveo Meraz

Submit a document that answers the following questions. If any question does not apply (e.g., if there are no major sections that are unclear), state that. BUT, it's highly unlikely that any document will be perfect. A review that does not include any suggestions will not receive credit (will be reflected in the Advisor Evaluation). Having the ability to review/edit/critique reports is a good skill!

1. All necessary sections were included: **Yes**

2. What is not clear? Either a) list subsection(s) of the document or b) cut-and-paste sentences or paragraphs that need work (you may also use Review option of Word/OO if you prefer)

**Test Items: Report using time constraints not filled out**

**Test Items: Triggers- has a TBD when this is the final report**

3. Where is more detail needed?

**The client section seemed rather shot, a bit more explanation would be appreciated since I have little background to this client.**

4. Any sections where verb tense was not changed from original document? (e.g., "we will do X" rather than "we did X"). Note that present tense is fine, especially if describing the design. OR where document does not flow (e.g., intermediate reports pasted together with no attempt to turn into cohesive document).

**In functional requirements, the first sentence should be in past tense since the project has ended. Non-functional requirements also has things in future tense.**

5. List any obvious grammar issues.

**In lessons learned, -- Also, there are very many Java libraries << sounds weird**

If you were grading this paper in its current form, what score would you give it? Fill in the rubric below. NOTE: only final submissions will be graded. The purpose of this is to give students feedback. If you would give 19/20, report is very close. If only 15/20, report needs some work.

| Item | Possible | Points |
|---|---|---|
| Document contains all the required sections. | 5 | 5 |

| | | |
|---|---|---|
| Document has adequate detail | 5 | 5 |
| Document is formatted correctly, including figures with labels (referenced in text) | 5 | 4 – figures within text not labeled |
| Document complies with style and grammar guidelines, no spelling errors | 5 | 5 |
| Total | 20 | 19 |

Team Reviewed: Salesforce 2

Reviewed by: Sunlight Photonics-Andrea Golden-Lasher

Submit a document that answers the following questions. If any question does not apply (e.g., if there are no major sections that are unclear), state that. BUT, it's highly unlikely that any document will be perfect. A review that does not include any suggestions will not receive credit (will be reflected in the Advisor Evaluation). Having the ability to review/edit/critique reports is a good skill!

1. All necessary sections were included: **Yes** No

2. What is not clear? Either a) list subsection(s) of the document or b) cut-and-paste sentences or paragraphs that need work (you may also use Review option of Word/ OO if you prefer)

**Overall question:** There is terminology (endpoints, gRPC) I am not familiar with, but maybe your client is. It may be helpful to the reader to define these terms early in the report.

**System Architecture-Description**: From 'There are two options, … to 'or check the test results' the wording is choppy

**Quality Assurance-Test Items:** Maybe consider summarizing the section and including the rest as an appendix? You cover a lot of good testing, but the deliverable for most of them is the same for all three pages.

3. Where is more detail needed?

**Quality Assurance-Test Items-Report Using Time Constraints:** Missing the bullet point information.

**Quality Assurace-Test Items:** Some descriptions are very vague or are still 'TBD'

4. Any sections where verb tense was not changed from original document? (e.g., "we will do X" rather than "we did X"). Note that present tense is fine, especially if describing the design. OR where document does not flow (e.g., intermediate reports pasted together with no attempt to turn into cohesive document).

**Requirements-both Functional and Non-Functional**: Add transitional sentence between the paragraph and the bullet points for better flow, such as "These requirements are summarized below:"

**Quality Assurance-Goals:** Add a sentence before the bullet points explaining what they are for/introducing the quality assurance section

5. List any obvious grammar issues.

**Cover Page:** Update the date to the day you turn it in

**Introduction-Client-Sentence 1:** worlds → world's

**Introduction-Client-Sentence 2:** comma after 'cloud-based'

**Introduction-Product Vision-Sentence 2:** developer's → developers

**Technical Design-Testing Endpoints:** figure missing proper format and in text reference

If you were grading this paper in its current form, what score would you give it? Fill in the rubric below. NOTE: only final submissions will be graded. The purpose of this is to give students feedback. If you would give 19/20, report is very close. If only 15/20, report needs some work.

| Item | Possible | Points |
|---|---|---|
| Document contains all the required sections. | 5 | 5 |
| Document has adequate detail | 5 | 5 |
| Document is formatted correctly, including figures with labels (referenced in text) | 5 | 4 |
| Document complies with style and grammar guidelines, no spelling errors | 5 | 3 |
| Total | 20 | 17 |

Team Reviewed: _____Salesforce 2_____

Reviewed by: _____Zachary Smeton_____

Submit a document that answers the following questions. If any question does not apply (e.g., if there are no major sections that are unclear), state that. BUT, it's highly unlikely that any document will be perfect. A review that does not include any suggestions will not receive credit (will be reflected in the Advisor Evaluation). Having the ability to review/edit/critique reports is a good skill!

1. All necessary sections were included: **Yes** No

2. What is not clear? Either a) list subsection(s) of the document or b) cut-and-paste sentences or paragraphs that need work (you may also use Review option of Word/OO if you prefer)

- Mostly in the usage of different vocabulary without definition or explanation. What are endpoints? What are endpoints responsible for? What does gRPC stand for?
- Quality Assurance Test Items is more of a useful document for the software developers than the final report reader, most of the language and information feels more like a testing plan than a write up about the testing process.

3. Where is more detail needed?

- System Architecture
- How an API is tested with this API
- Requirements are short and sweet but uses technical terms without explanation (RESTful, API contract)

4. Any sections where verb tense was not changed from original document? (e.g., "we will do X" rather than "we did X"). Note that present tense is fine, especially if describing the design. OR where document does not flow (e.g., intermediate reports pasted together with no attempt to turn into cohesive document).

- Requirements is not in present or past however that feels correct to me… They were the goals for the project, so future seems right.
- Quality Assurance – Goals (for lack of flow), Strategy, Test Items (contains some unimplemented requirements), Risks, Results of usability tests

5. List any obvious grammar issues.

Salesforce is the company behind the world's #1 customer relationship management solution.
Their software is cloud-based, making it easy for teams to setup and use.
The design of our Mock API service can be shown using two flow charts, see Figure 1 and 2 of the appendix.
There are two options, their software will generate a request which will be sent to our Mock API. <- just not a sentence

Mixed usage of capitalization for the names of different API calls like “API test call” and “Results call.” Any request that is not captured in the endpoints that we have specifically created is checked to see if it is a user created endpoint.

If you were grading this paper in its current form, what score would you give it? Fill in the rubric below. NOTE: only final submissions will be graded. The purpose of this is to give students feedback. If you would give 19/20, report is very close. If only 15/20, report needs some work.

| Item | Possible | Points |
|---|---|---|
| Document contains all the required sections. | 5 | 5 |
| Document has adequate detail | 5 | 4 |
| Document is formatted correctly, including figures with labels (referenced in text) | 5 | 3 |
| Document complies with style and grammar guidelines, no spelling errors | 5 | 4 |
| Total | 20 | 16 |

Team Reviewed: Salesforce

Reviewed by: Josh Nachtigal

Overall: Your report already seems pretty complete, but I honestly don't really understand the details of the project. It may make sense to your client which is most important, but some more detail may be necessary for the advisors and other people reading the report.

1. All necessary sections were included: Yes!

2. What is not clear? Either a) list subsection(s) of the document or b) cut-and-paste sentences or paragraphs that need work (you may also use Review option of Word/OO if you prefer)

- gRPC is refenced many times but as far as I could see (I could have missed it) it is never defined.

- also RESTful, acronyms probably be defined on first use unless very well known

- in the system architecture paragraph (pg 4) the test results are referenced a couple times, but the format is not specified here. Is it a HTTP/JSON file like the endpoints?

- in performance testing results (pg 11) does 96% of code covered mean 96% of code resides in some unit test or does this have some other meaning?

3. Where is more detail needed?

- The results section feels slightly short but given that most tests seem to be unit tests which are pass/fail this may be fine. If possible, this could be filled out more

4. Any sections where verb tense was not changed from original document? (e.g., "we will do X" rather than "we did X"). Note that present tense is fine, especially if describing the design. OR where document does not flow (e.g., intermediate reports pasted together with no attempt to turn into cohesive document).

-None that I found

5. List any obvious grammar issues.

- Many statements are not clear or grammatically correct. For example, in the system architecture paragraph, 'There are two options, their software will generate a request which will be sent to our Mock API.' does not really make sense. This could likely be resolved though a read through.

| Item | Possible | Points |
|---|---|---|

| | | |
|---|---|---|
| Document contains all the required sections. | 5 | 5 |
| Document has adequate detail | 5 | 3 |
| Document is formatted correctly, including figures with labels (referenced in text) | 5 | 5 |
| Document complies with style and grammar guidelines, no spelling errors | 5 | 4 |
| Total | 20 | 17 |

Team Reviewed: Salesforce 2

Reviewed by: Kristin Farris

Submit a document that answers the following questions. If any question does not apply (e.g., if there are no major sections that are unclear), state that. BUT, it's highly unlikely that any document will be perfect. A review that does not include any suggestions will not receive credit (will be reflected in the Advisor Evaluation). Having the ability to review/edit/critique reports is a good skill!

1. All necessary sections were included: Yes No

2. What is not clear? Either a) list subsection(s) of the document or b) cut-and-paste sentences or paragraphs that need work (you may also use Review option of Word/OO if you prefer)

- Product Vision: After reading this section, I am still a little confused about the goal of the project. Be a bit more descriptive and detailed on what "to create an API that could be used in the testing of other APIs" means.
- Overall, what is meant by "requests"- http requests? What's a gRPC request?
- System Architecture: I feel like including figure 1 in the main part of the report instead of the appendix would help with comprehension. Its not too big so I don't think it would be distracting
- System Architecture: To help with readability I would recommend "If the API receives a new endpoint call..." → "If the API receives a 'new endpoint' call..." or "If the API receives a *new endpoint* call..." for that sentence and others that talk about specific calls
- Technical Design: I think if things were reworded to be more concise in this section, it would make more sense. For example "Any request that is not captured in the endpoints that we have specifically created is checked to see if it is user created endpoint. This is done by checking the path and method of the request against our list of user created endpoints." could be easily reduced to "Any request not captured in our predefined endpoints is checked against a list of user-created endpoints"
- Results: "we were able to cover 96% of all lines in the program" elaborate on what cover means because it's a little confusing
- Appendix: The second figure is too blurry to read

3. Where is more detail needed?

- Functional Requirements: “accept certain types of requests”, “able to respond with specific responses” very vague
- Non-Functional Requirements: maybe make the the bullet point list more sentence-like- for example Java 8 → Written using Java 8
- Figure labels needed
- Quality Assurance: Perhaps lead into this section with a few sentences, starting out with the goals bullet pointed is abrupt. The same goes for the start of the Test Items subsection
- Test Items: there is nothing under the “Report Using Time Constraints” test
- Results: In the introduction it was stated “We have implemented all of the client’s required features as well as many of the other stretch goals” but here “There were some features that we were not able to implement due to time constraints”. More detail on which one it is is needed
- Lessons Learned: I think elaborating on these points by making them actual paragraphs with fully proper grammar would be helpful

4. Any sections where verb tense was not changed from original document? (e.g., “we will do X” rather than “we did X”). Note that present tense is fine, especially if describing the design. OR where document does not flow (e.g., intermediate reports pasted together with no attempt to turn into cohesive document).

- Product Vision: “the service needed” in one sentence and “the service also needs” in the next. Pick one tense
- Quality Assurance: “However, as we add new features, we will also need to perform integration and progressive regression testing” You should be done with adding features, so make this past tense
- Results: “The client is happy with the product so far, although there is still work that could be done to improve it”- “so far” doesn’t make sense with the project being over

5. List any obvious grammar issues.

- Introduction: worlds → world’s
- System Architecture: “There are two options, their software will generate a request which will be sent to our Mock API.” This sentence is not a sentence.
- System Architecture: Grammar and capitalization is kind of all over the place in this section.

- Capitalization standard of “Spring”, “Java” and “Mock API” changes throughout
- Overall, I think a read through of all the sections is needed. A lot of the grammar mistakes and strange wording throughout would be easily fixed with another read through.
- (More format than grammar) Pick an indentation standard, it changes towards the end

If you were grading this paper in its current form, what score would you give it? Fill in the rubric below. NOTE: only final submissions will be graded. The purpose of this is to give students feedback. If you would give 19/20, report is very close. If only 15/20, report needs some work.

| Item | Possible | Points |
|---|---|---|
| Document contains all the required sections. | 5 | 5 |
| Document has adequate detail | 5 | 4 |
| Document is formatted correctly, including figures with labels (referenced in text) | 5 | 4 |
| Document complies with style and grammar guidelines, no spelling errors | 5 | 3 |
| Total | 20 | 16 |

## Addressing Feedback

### Alejandro Caraveo Meraz

All feedback addressed except for past tense in requirements because they are still required.

### Andrea Golden-Lasher

All feedback addressed.

### Zachary Smeton

This is a technical document so the reader should know most of the vocabulary. The testing information was moved to the appendix. More detail has been added in most places, including system architecture. API testing is explained in the technical details section. Grammar was fixed in the places specified.

### Josh Nachtigal

The reader should know most technical terms. Everything else has been addressed.

### Kristin Farris

We addressed pretty much everything. Did not address the request for an introduction section for “quality assurance” section since we feel the “goals” summary gets the job done.