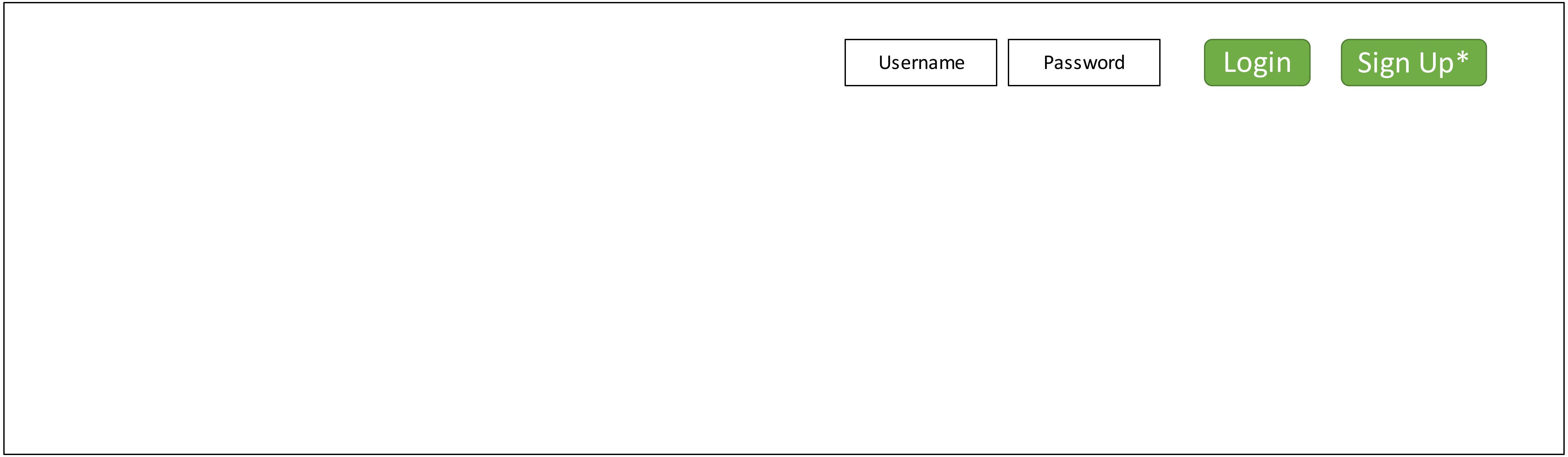
GDIS Test Suite

System Design, Required features and sample

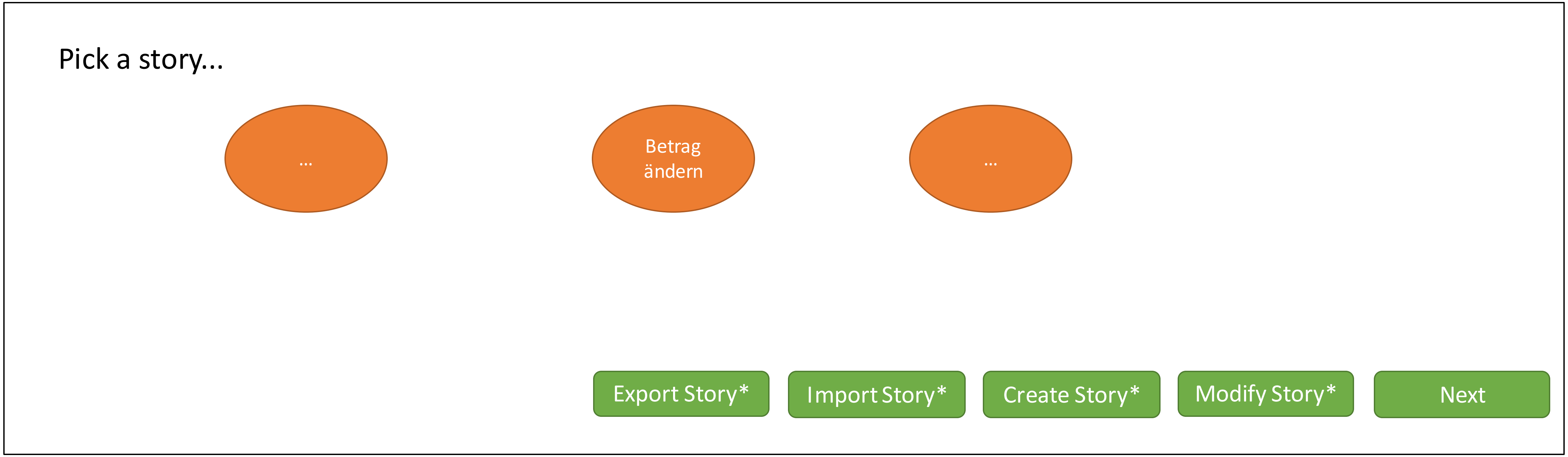
index.html



- Optional Login
- Index page can be used for login only and to redirect to the story picker page or to display stories after login

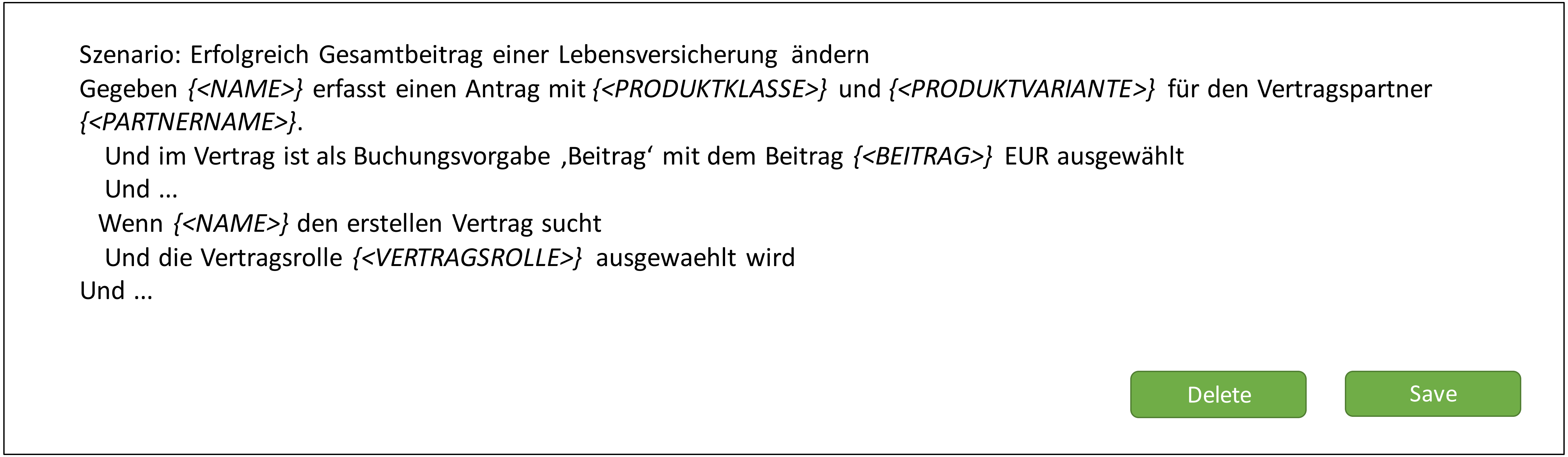
* = Option
(L) = implement in late stage

story_picker.html



- Select an existing story and proceed to the Test Data Manager
- Modify Story Option (Including delete)
- Create Story (Including import) and save to .story file on the server; Go to story modifier page after import
- Export story to file (download option) or to Git (open new dialog on the same page)

story_modifier.html / story_manager.html *



Szenario: Erfolgreich Gesamtbeitrag einer Lebensversicherung ändern

Gegeben *{<NAME>}* erfasst einen Antrag mit *{<PRODUKTKLASSE>}* und *{<PRODUKTVARIANTE>}* für den Vertragspartner *{<PARTNERNAME>}*.

Und im Vertrag ist als Buchungsvorgabe ‚Beitrag‘ mit dem Beitrag *{<BEITRAG>}* EUR ausgewählt

Und ...

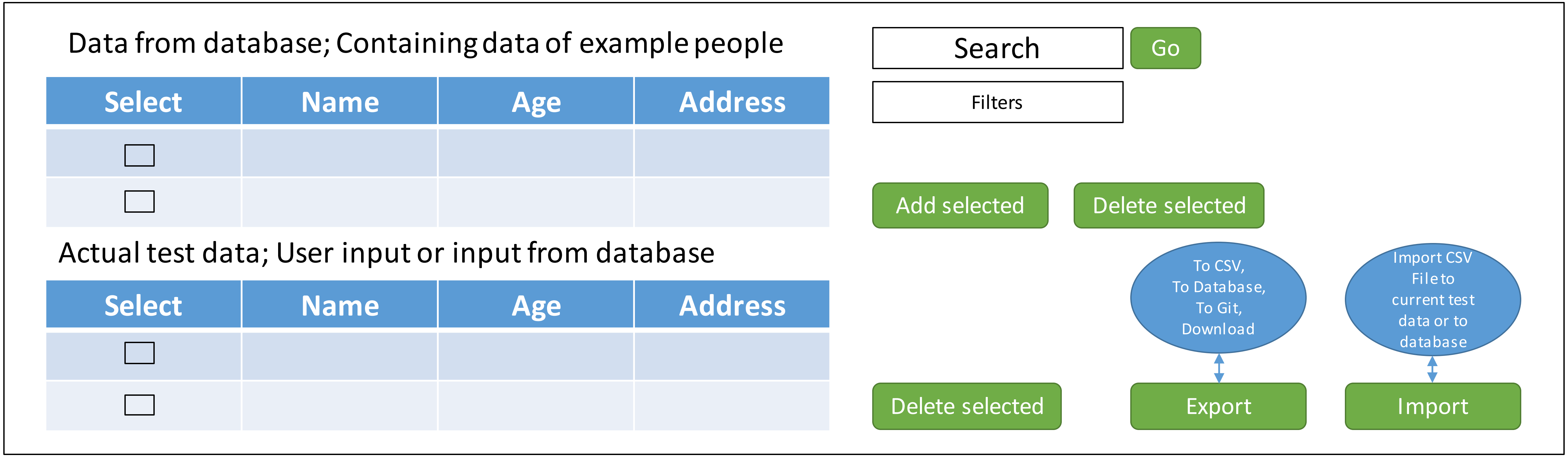
Wenn *{<NAME>}* den erstellen Vertrag sucht

Und die Vertragsrolle *{<VERTRAGSROLLE>}* ausgewaehlt wird

Und ...

- Land on this page from the Story Picker Page
- Use it for story creation and modification
- Add new field for every action and keep only one field for both “Szenario” and “Gegeben”

test_data_manager.html



Data from database; Containing data of example people

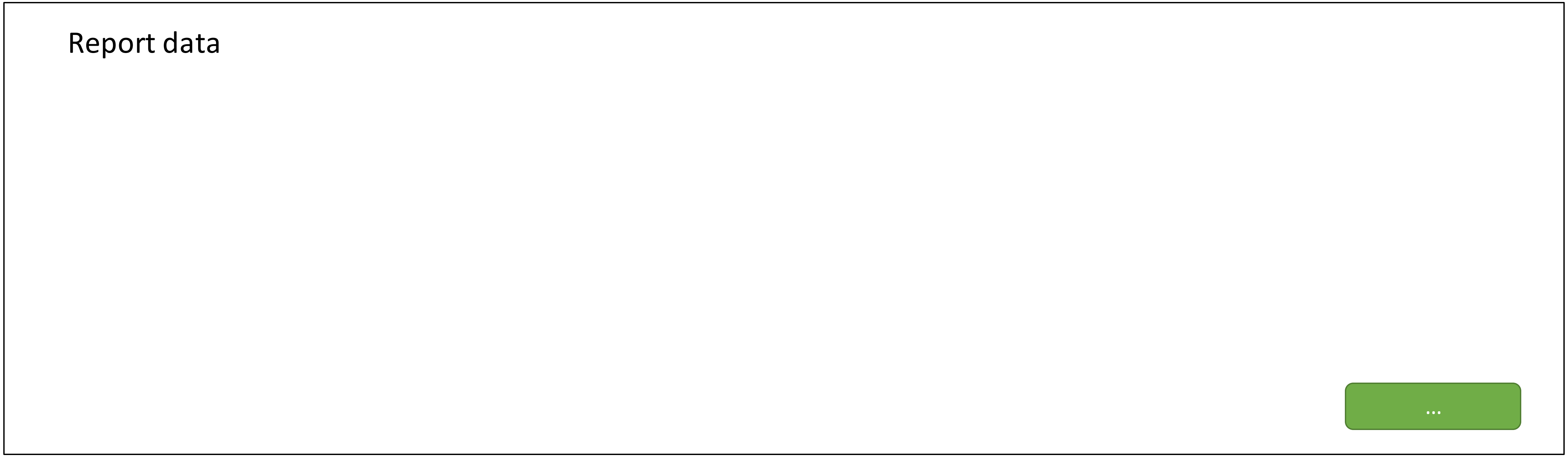
Select	Name	Age	Address
□			
□			

Actual test data; User input or input from database

Select	Name	Age	Address
□			
□			

- The top table should display example users which are saved in the database or already saved test cases; “Add Selected” to add a field to current test’s data, note: Saved entry might have more/less attributes (columns) than the current test requires; “Delete Selected” to delete entry from DB
- The bottom table should display the test data for the current test
- The fields in both tables should be modifiable and changes should occur live without reloading; Tables should be dynamic (add/remove columns/rows without reloading)
- Live search for users, saved tests, etc. with the help of filters
- Import test data to database or to the current test data from a CSV files
- Export test data to database or to CSV file with a download option or Git

test_result_dashboard.html (L)



Report data

...

Frontend technologies

- HTML/CSS/JavaScript
- AngularJS or Reacts
- Bootstrap or Foundation for Responsive design
- AJAX for live table management

Backend technologies

- Spring Boot
- MySQL or/and MongoDB Database for saving test data and dummy user data; Also for login information
- Search: ElasticSearch
- Docker Image containing only the Spring Boot project; Database on separate server

Microservices

- Test Data Importer:
  - Import data from CSV files to database
  - Create table for every CSV file ???
  - Import from link (Git)/Upload in browser/Browse a file from the server's file system (parts of it) *
- Test Data Exporter:
  - Create CSV file for a given test case
  - Save test case from browser to database
  - Export to Git/Download CSV file *
- Test Data Manager:
  - Manage Exporter and Importer and Handle AJAX/REST Requests
- Test Data Generator: ???
- CSV File Manager:
  - Check Syntax of files
  - Create Files

Microservices

- Search (L):
  - Implement after the database is set up
- Test Result Importer (L):
  - Manage Jenkins reports
- Analytics (L)
- Test Runner (L)
- Story Manager (L):
  - Create, modify and delete Stories

Database

- MySQL or MongoDB; SQLite also possible
- Probably using SQL will be easier since we are working with CSV files
- Database structure/schema TBD
- MongoDB will be great for REST, but CSV <-> JSON might introduce bigger overhead
- Concurrent access later; Initially simple lock or nothing

REST

- GET: Get test data/CSV File in JSON
- POST: Insert test data in the Database/Generate CSV file on Server from JSON; possible for stories too
- DELETE: Delete table, test data, row from table, etc.

Login

- MySQL or MongoDB; SQLite also possible
- Hash passwords
- User information TBD
- Forgotten Password feature
- Sign Up
- User privileges

Security

- SQL Injection, XSS, XRF (Cross site request forgery)
- Session Management for Login
- Token Management for REST API
- Hash Passwords
- Filter all inputs while generating test data (don’t allow text in a number field, etc.), RegEx
- Handling error codes 404, 403, etc.
- Possible encryption ???