**Introduction:**

Our client, Anonymous Builders (AB), provides residential construction services to include outbuildings, decks, and light interior renovations. To assess worksites, take measurements, and generate estimates for potential customer’s, AB must dispatch estimators to residences. Upon completion of their tasks, these estimators record their findings and projected costs, along with customer information (name, phone number, email, etc.) on paper forms. Frequently, estimators use their mobile devices to take photos of the job site.

After returning to AB’s office, the estimators manually input the data from their paper forms into a spreadsheet maintained on a shared network drive. Photos are also transferred from the mobile devices to the network storage. This data is used for vital processes including recording project details, maintaining future contact with customers, and studying business trends. AB management periodically backs ups this information to a Google Drive account.

Their existing system has encountered several problems, to include:

* Paper forms going missing, being illegible, or being damaged prior to spreadsheet entry.
* Estimators delaying data entry, creating a backlog of unrecorded paper forms.
* Data duplication, deletion, or corruption within the spreadsheet file.
* Certain technically non-proficient estimators having difficulty with filesystem navigation, photo transfer, and spreadsheet manipulation.
* Ongoing concern with on-premise data security and loss.

Wishing to resolve these problems, which are detrimental to the AB’s productivity, the company has hired us to create a software solution that addresses all the above while fitting in with their existing hardware infrastructure. Their strategic objective is to see an eventual ROI from this project’s expenditure by reducing the office labor overhead created by their existing system, resulting in estimators spending more time in the field collecting data from prospective customers.

A secondary aim of this software, which also contributes to the stated objective, is to reduce the period of delay in which office staff do not have access to customer info, resulting in quicker follow-up with prospective customers and, potentially, expedited scheduling of work crews.

**User Requirements Definition:**

AB seeks a complete replacement of their existing customer/project data system. AB management is asking us to develop a system that meets the following requirements:

* Switch the assessment/estimation to a primarily digital model, making paper documentation completely optional
* I.e., initial entry could be directly digital, though paper reports could be printed afterward as needed for distribution or backup filing
* Secure, primary storage of all data off-site from the office and only accessible to authorized personnel
* Ability to update stored data in the field without necessitating a return to the office
* Ability to upload job site photographs remotely
* Interface accessible from both company PCs or personal mobile devices. If browser interaction is required, it must be compatible with Chrome and Firefox browsers.
* AB has very few employees who use iOS devices; they will provide Android tablets to these. AB has determined the extra cost of Safari compatibility testing is not of value to them.
* AB has also decided to forgo testing for other browsers such as Internet Explorer, Edge, Opera, etc. to lower development time and costs
* The system must not require any additional hardware. For example, since AB has no dedicated IT staff, the company has no interest in purchasing nor maintaining their own physical database server.
* Interface must be easily utilized by non-technical personnel. End-users will be estimators, office staff, and work crew foremen with varying degrees of technical proficiency.
* Easy storage and retrieval of stored data to allow uninhibited business operations
* AB is not seeking to have us migrate their existing paper or digital records into the new application. Under the terms of this current project, any migration of old data will be conducted by their own staff after delivery of the final product.

*Diagram

Description automatically generated*

* Data Requirements:

To provide a straightforward representation of the specific data collection requirements needed by our client, a sample “Create Customer Record” form is provided below, which represents one of the functional “pages” of the application. This provides, at a glance, the fields of data an AB estimator would require for properly adding a new customer, along with an assessment of the job associated with their initial consultation, into the company’s records.

*Please note that this mock-up is from an unstyled HTML document that is not intended to represent the actual application browser UI in any sense beyond data requirements.*

AB employees also require the ability to retrieve, view, edit, and delete existing records via the application’s UI. The client requires that these processes be implemented in a fashion that minimizes time spent on redundant data entry. For example, a new customer record should not need to be created for a customer that AB has performed a job for previously; the application must have the ability to associate a new job record with an existing customer record.

Graphical user interface, application

Description automatically generated

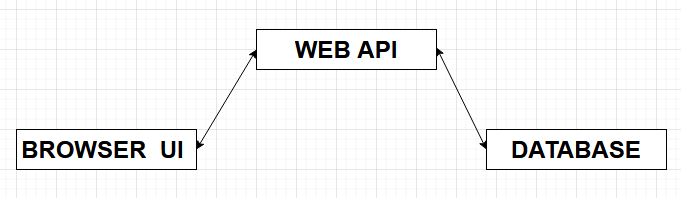
**System Architecture:**

Our initial assessment of the requirements indicate a web-based application would be the solution most appropriate to meet both AB’s usability/hardware/functionality requirements, the capabilities of our team, and the time/budget constraints of the project. By having the application be compatible with popular browsers Chrome and Firefox, this will ensure that the office PCs as well as many mobile devices should be able to interact with the application. This will also help our team meet the project deadline since we will not have to develop multiple native applications for various platforms nor perform testing on a variety of browsers. The low system resource and bandwidth demands of the application should make performance-focused architecture and testing relatively low-priority concerns.

The application will consist of the following primary components:

* A database with these features:
  + Stored on a secure, reputable cloud platform
  + Ideally, this would use a “serverless” or similarly simple interface that would require little or no managing on the part of AB’s staff
  + Access via username/password authentication
  + Relational structure so that multiple job records may be associated  
    with the same customer record
  + CRUD functionality
* Browser forms with these features:
  + Simple, easy-to-use GUI display upon page load
  + Reactive to user screen size (mobile or PC)
  + Form pages for record creation, customer data entry, job data entry, and queries/management
* An API to allow the browser front-end to interface with the database on the back-end

The structure diagram below represents the union of the three core components of the application that were outlined above.



The below diagram represents the flow of data through the application’s components. This is an abstraction of the data paths and not indicative of final naming of variables, etc.

Diagram

Description automatically generated

The software architecture will follow a Model-View-Controller (MVC) design pattern, represented by the data passed to/from the database (Model), the browser record display (View), and the interactive forms elements that allow manipulation of the data and display (Controller).

The code will follow an object-oriented, class-based design with individual customer records, job records, and display views represented by instantiations of corresponding classes. The customer and job objects will correspond to matching records in the appropriate database tables. Records being built for or retrieved from the database will be represented by display view objects; these view objects are not persistent and are only for purposes of browser display rendering.

Given the ubiquity of this design and of some of the classes included, we have determined that such an architecture has high potential for reuse of code components in future projects with AB or other clients. As a widespread and time-tested design pattern, it also meets high standards of elegance and readability that should help ensure easy adoption by the development team.

**System Evolution:**

Future evolution of the system will be primarily based on feedback from end-users at AB, including office management, field estimators, and work crew foremen. Improvement areas will include ease-of-use, security, and expanded features. Since the initial design is implemented in a very modular and system-agnostic manner, the project could eventually evolve to include desktop or native mobile application front-ends which could interface with the existing database.

**Appendices:**

*Hardware requirements:*

* AB’s existing PCs
* The personal mobile devices of AB’s employees (various smartphones, tablets, etc.)
* AB intends to provide basic Android tablets to employee who do not already own a suitable Chrome or Firefox compatible device
* Off-site cloud servers provided by third-party hosting service, *Amazon Web Services*

*Proposed software requirements:*

* AWS services
* DynamoDB database
* Lambda
* API gateway
* Authentication
* Firefox and Chrome browser installations
* Browser UI
  + - Option 1: Create using raw HTML, CSS, JavaScript
    - Option 2: Investigate possible frameworks like Vue.js which might enhance and/or expedite front-end development
* An API script that provides for an interface between the browser UI and the database

*Languages used:* We propose to develop the browser UI using the aforementioned trio of web languages. JavaScript with Node.js runtime has been chosen as the development language for our API. Some SQL may also be needed depending on how we configure our data requests.

**Systems Development Life Cycle:**

*Requirements Analysis and Definition:* Reassess to ensure Requirements as stated above will be met by the Architecture outlined. Develop user stories and present them to a variety of end-users at AB to gather more accurate usage cases and expectations. Incorporate any feedback from the client after their review of the relevant portions of this document.

*System and Software Design:* Determine which Proposed Software Requirements will be utilized. Delineate the exact data to be collected in the web forms, then ensure database tables are constructed appropriately. Ensure API endpoints information is established. These steps will ensure the team can work on the modules independently with minimal misalignment of connection points and code rework.

*Implementation and Unit Testing:* Ensure proper functionality of database (authorization, CRUD functions). Test web form for proper interactions. Ensure API is successfully transferring information.  
A variety of possible inputs must be tested to ensure that input validation code is not allowing user input that could cause errors or present security risks.

*Integration and System Testing:* Test complete system on mobile and PC platforms in multiple browsers. Particular points of interest would be screen size/device responsiveness, input device compatibility (touchscreen, keyboard/mouse, etc.), and proper interaction of the photo upload feature with device photo storage.

*Operation and Maintenance:* Collect feedback, fix bugs, and evaluate requirement/feasibility of additional features.