Deep.Ditch

Software Requirements Specification

Version 1.0

10/09/2018

Drake Svoboda

Hassan Bazzi

Safayeth Khan

Revision History

Date	Description	Author	Comments
10/9/2018	Version 1.0	Drake Svoboda Hassan Bazzi Safayeth Khan	First Draft

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Drake Svoboda	Team Lead/ Machine Learning Lead	10/09/18
	Hassan Bazzi	Web Application Lead	10/09/18
	Safayeth Khan	Mobile Application Lead	10/09/18

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe—in detail—the software requirements for the deep.ditch platform. This document will act as a contract between the client and developers and contains information that pertains to the agreed upon features that will be implemented. This document provides a detailed description of these features and will cover how all of the required technologies, hardware, and software will interact.

1.2 Project Scope

The deep.ditch platform consists of two applications: a web application and a mobile (iOS) application. The mobile application will contain a machine learning model to detect road damage in images captured by the device. Road damage identified by the model will be reported by the mobile application to the web application's API. The platform will maintain a data-store of reported road damage and make it available via an API and a user interface.

1.3 Definitions, Acronyms, and Abbreviations

User, General User	The person who uses the application and is not an organization administrator.
Admin, Administrator, Organization Administrator	A user with an organization administration role. This role has special permissions.
External Software Client	Any third party system that wishes to consume the web application's exposed REST API.
Precision	(true positives) / ((true positives) * (false positives))
Recall	(true positives) / ((true positives) * (false negatives))
F-Measure	2 * (precision * recall) / (precision + recall)
Mobile device	The device on which the mobile application will be running on.
Damage instance	An instance of road damage in our database which we have detected using our platform.

1.4 References

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images. *Computer-Aided Civil and Infrastructure Engineering*. doi:10.1111/mice.12387

1.5 Overview

This document is to describe in detail the software requirements for the deep.ditch platform. Thus, this document will provide an overall description of the product. The overall description has information that pertains to product perspective, product features, user classes and characteristics, design and implementation constraints, and general assumption. The document then enumerates the specific requirements of the platform and describes how each of its components interact.

2 Overall Description

2.1 Product Perspective

In order to effectively maintaining their roadways, government officials must be aware of the locations and types of road damage within their jurisdiction. Currently, the process of of road damage reporting is either manual and inefficient or automated and cost-prohibited. Governments rely on manual inspections, public reports, or expensive equipment to report road damage. deep.ditch provides a platform for road damage reporting that is both automated and cost-efficient. Additionally, our proposed platform will not only identify road damage, but also classify road damage into eight categories (Table 1.).

2.2 Product Features

The purpose of this software is aid municipal organizations with road maintenance by providing an automated solution for road damage reporting. We will train a machine learning model on a dataset of 7,240 cell-phone images to identify and classify road damage into eight categories.

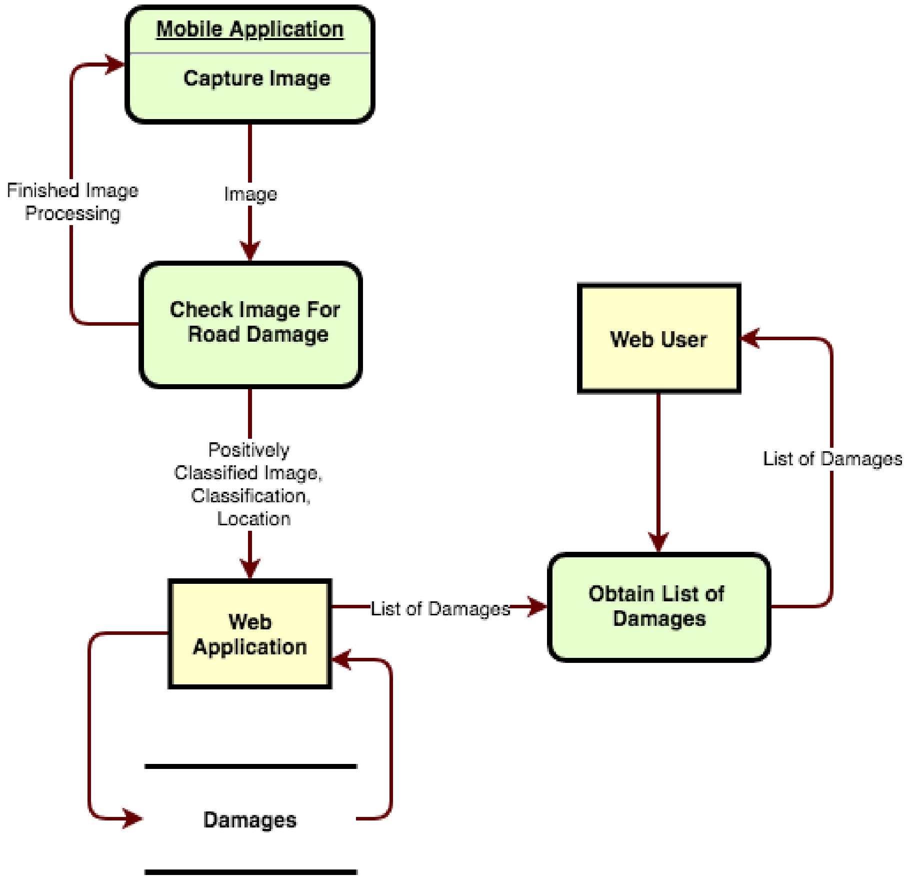
Table 1. *Road damage types and definitions*

Damage Type			**Detail**	**Class Name**
Crack	Linear Crack	Longitudinal	Wheel mark part	D00
			Construction joint part	D01
		Lateral	Equal interval	D10
			Construction joint part	D11
	Alligator Crack		Partial pavement, overall pavement	D20
Other Corruption			Rutting, bump, pothole, separation	D40
			White line blur	D43
			Crosswalk blur	D44

Note. Reprinted from *Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone,* by Maeda et al., retrieved from arXiv:1801.09454.

A mobile device running our model could then be mounted to the dashboard of a government vehicles so that the roadway is in clear view of the device's camera. The mobile device would then process images from the camera in real time as the vehicle is moving. When road damage is detected in an image, the location of that damage will be recorded by the device delivered to a web API. The web application will store the data and provide a user interface and expose an API for fetching the reported road damage. Government officials will be able to see road damage within their jurisdiction and take appropriate action.

Figure 1. *Minimum data flow diagram*



Note. See section A.1. for a complete data flow diagram.

2.3 User Classes and Characteristics

The deep.ditch platform is intended for use by government officials tasked with maintaining road ways. There are three general classes of users of the deep.ditch platform.

2.3.1 General User

A "General User" would be the person operating the vehicle on which the mobile device running the deep.ditch mobile application is mounted. They will have enough permission to run the application and allow the dash mounted device to collect the data.

2.3.2 Organization Administrator

An "Organization Administrator" would be the 'manager' of a group of general users (an organization). They will have permissions for a variety of additional features regarding organization and user management, along with all of the permissions of a general user.

2.3.3 External Software Client

An "External Software Client" is any third party system that wishes to consume the web applications exposed REST API. These third parties will require an API token generated and provided by an organization administrator (3.2.11). These third parties will have access to reported road damage so that data recorded by our platform could be easily integrated into existing systems.

2.4 Assumptions and Dependencies

2.4.1 Web Application

It is assumed that users of the web application will be using one of the following browsers: Internet Explorer 10+, Microsoft Edge 17+, Google Chrome 68+, Mozilla Firefox 61+, Safari 11. The user interface may have an inconsistent appearance across these browsers. Some functionalities may be unavailable in older versions of these browsers.

2.4.2 Mobile Application

It is assumed that the mobile application will be used on a device running iOS 11 or greater. The device should have available internet and location services and a functional camera.

It is assumed that—given current state-of-the-art machine learning techniques—the target mobile device will be able to efficiently run a machine learning model of adequate complexity i.e. a model that achieves non-functional requirement 3.6.1.

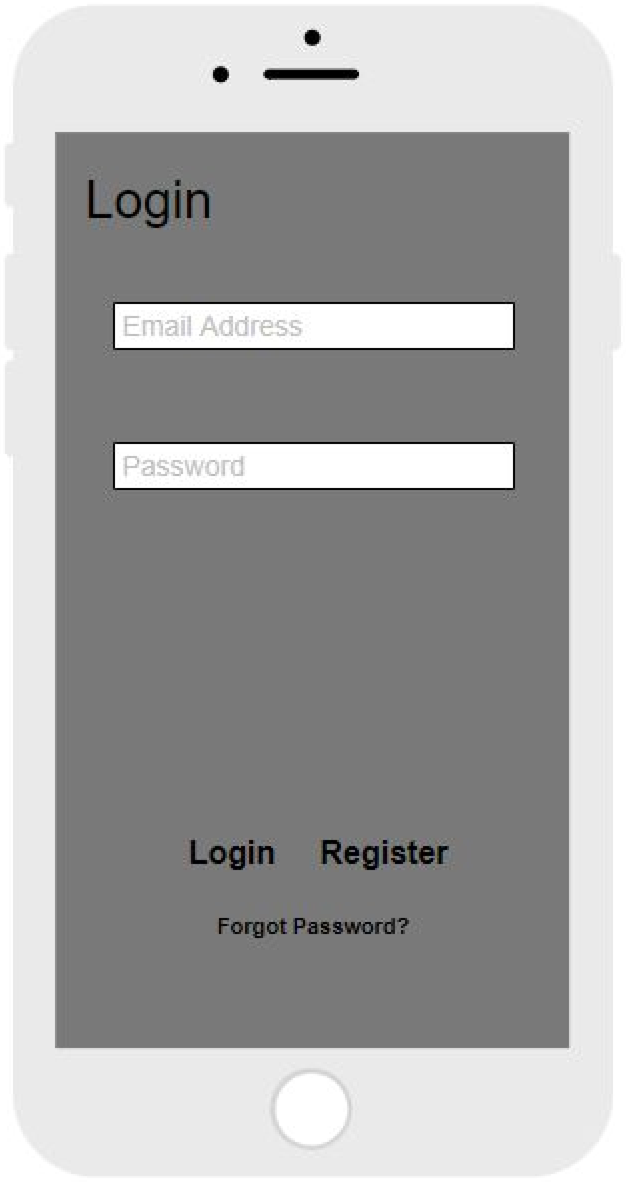
3 Specific Requirements

3.1 Interface Requirements

3.1.1 Mobile Application User Interface

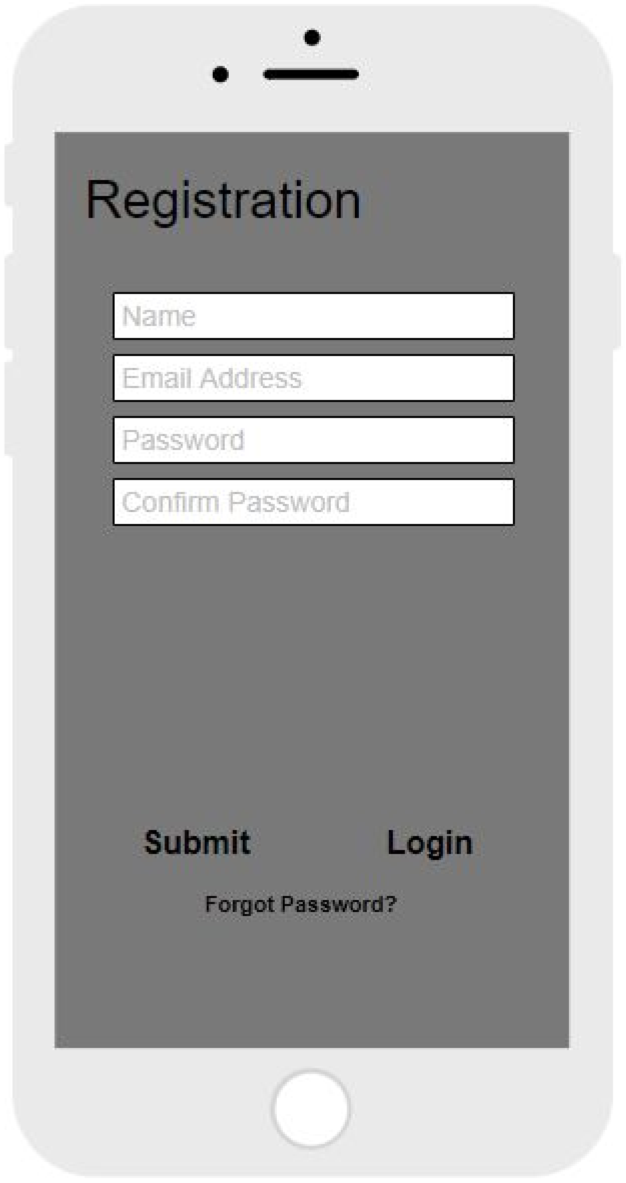
3.1.1.1 Login Screen

When the app is opened, it will open up to the login screen. If the user is already registered, they can login by filling in the email and password fields and pressing the login button. If they are not a registered user then they have the choice of registering using the register button, which will take them to the registration page. The user will be also have access to a forgot password button to recover the password.



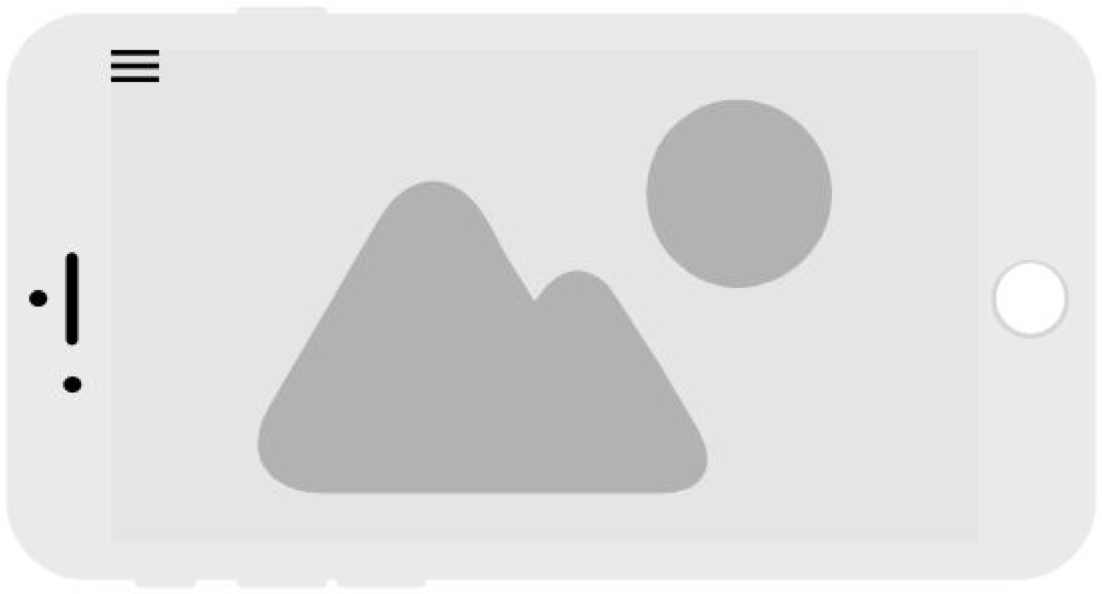
3.1.1.2 Registration Screen

In the registration screen, the user will be presented with inputs for name, email, and password. After the user enters their information, they will be able to click submit and the web application will register them. Upon a successful registration, the user will be redirected to the login screen.



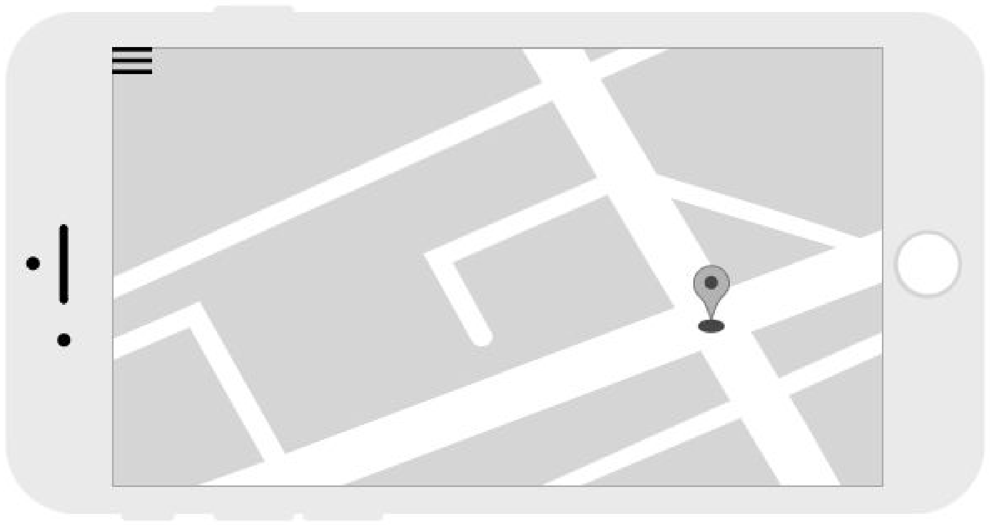
3.1.1.3 Camera Screen

After a successful login, the user is taken to the camera screen. In the camera screen, images from the devices camera will be processes in real time by the natively running machine learning algorithm and road damage will be automatically reported to the web API. The user will be presented with a view finder for the camera to aid in mounting the device on a vehicle's dashboard.



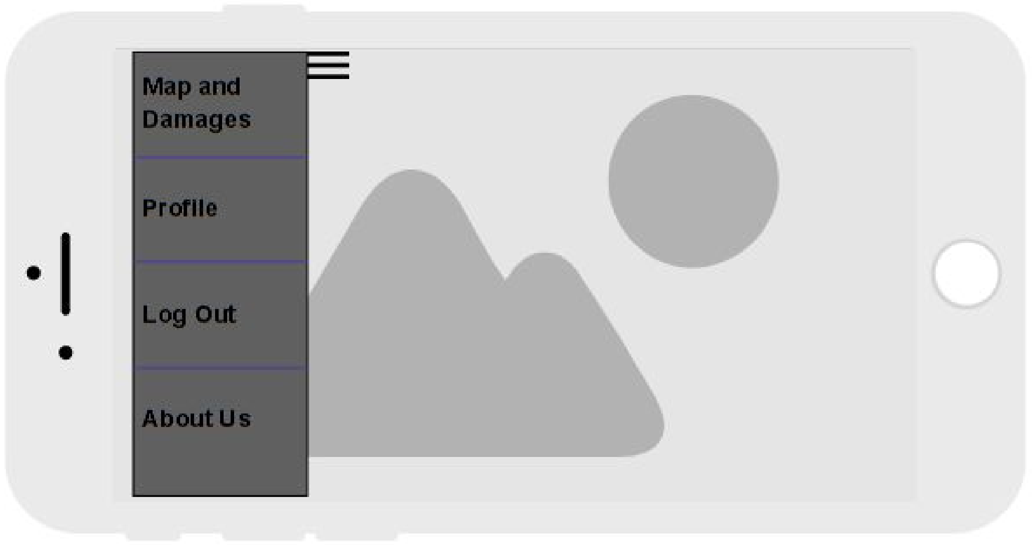
3.1.1.4 Map Screen

In the map screen, the user will be able to see road damage reported in their area. From this screen, the user will be able to access individual road damage instances and view their associated pictures.



3.1.1.4 Menu Button

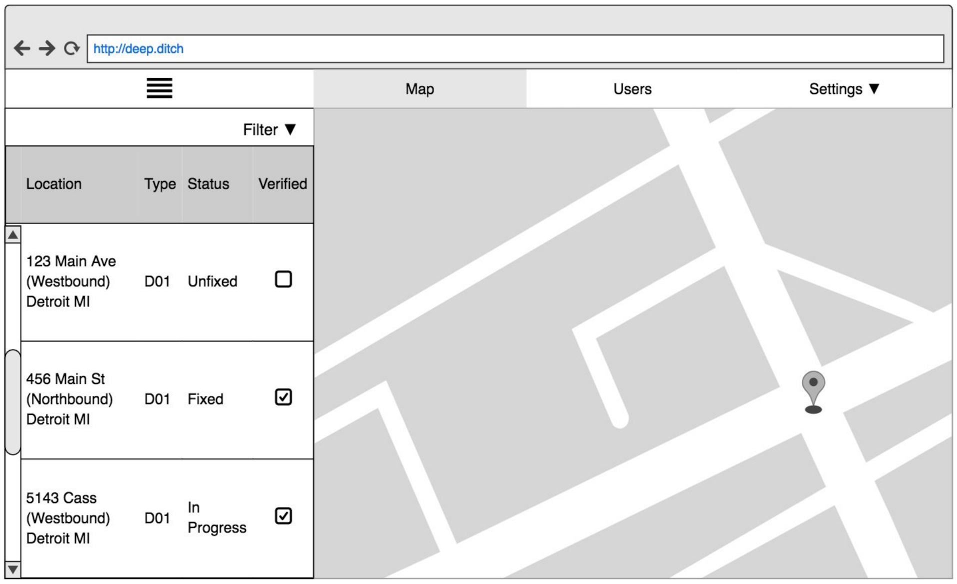
There will be a menu button available on the screen that will expand a menu. This menu will allow the user to navigate between the camera screen and the map screen. They should have the option to logout here. The logout button will take the user back to the login page. They also have the options for a profile page, where they can edit their profile, change password or the email that is associated with the their account.



3.1.2 Web Application User Interface

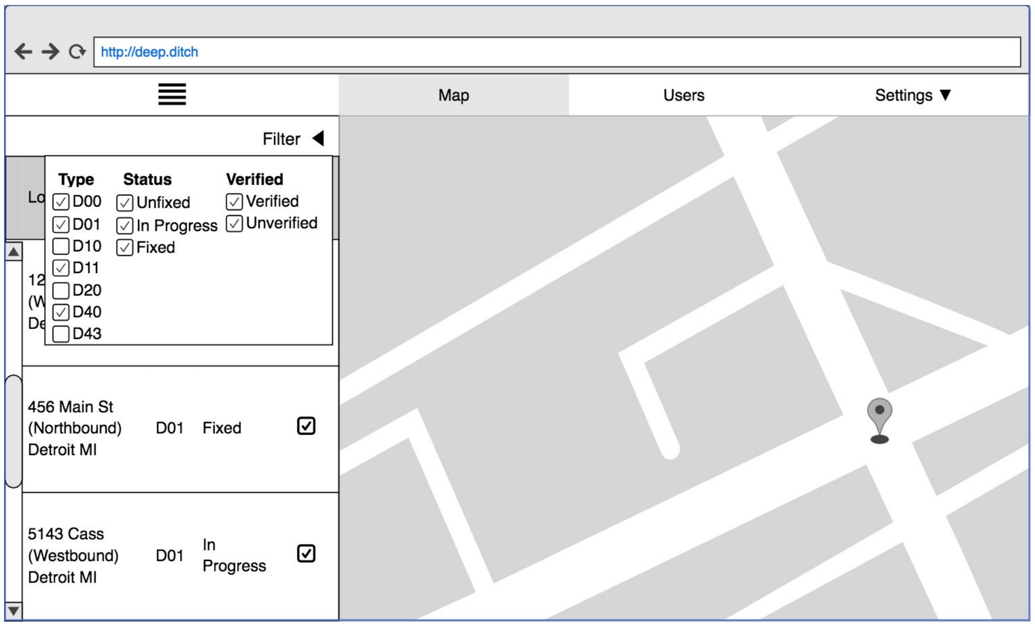
3.1.2.1 Map and List of Road Damage Entries

Upon login, there will be a Map and List view. The map will show pins with the exact locations of reported road damage.



Location	Type	Status	Verified
123 Main Ave (Westbound) Detroit MI	D01	Unfixed	☐
456 Main St (Northbound) Detroit MI	D01	Fixed	✓
5143 Cass (Westbound) Detroit MI	D01	In Progress	✓

The list on the left will show data about the locations on the map. The list is also filterable. What appears on the list will determine what pins appear on the map on the right.

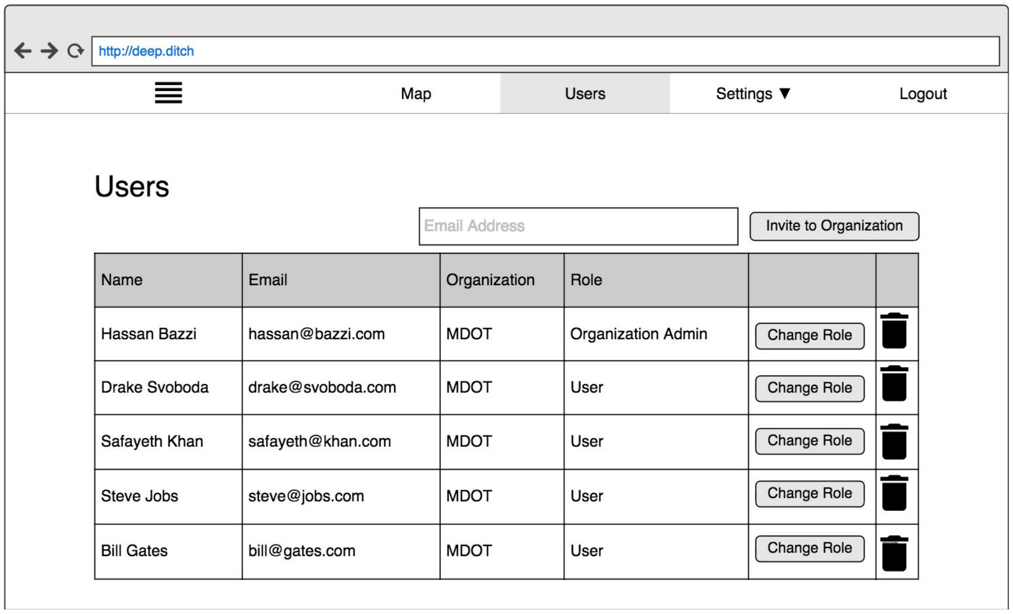


Type	Status	Verified
✓ D00	✓ Unfixed	✓ Verified
✓ D01	✓ In Progress	✓ Unverified
☐ D10	✓ Fixed	
✓ D11		
☐ D20		
✓ D40		
☐ D43		

456 Main St (Northbound) Detroit MI	D01	Fixed	✓
5143 Cass (Westbound) Detroit MI	D01	In Progress	✓

3.1.2.2 Users

The users tab will only be exposed for organization administrators to view, delete, and change the roles of the users in the organizations. Organization administrators will also be able to invite users to the organization.

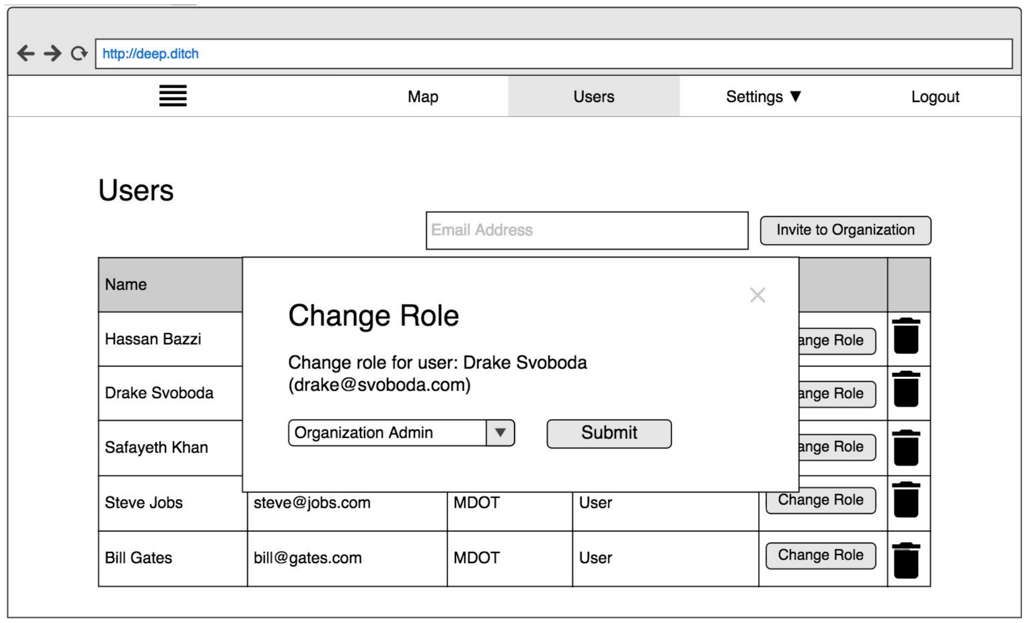


Users

Email Address | Invite to Organization

Name	Email	Organization	Role		
Hassan Bazzi	hassan@bazzi.com	MDOT	Organization Admin	Change Role	🗑
Drake Svoboda	drake@svoboda.com	MDOT	User	Change Role	🗑
Safayeth Khan	safayeth@khan.com	MDOT	User	Change Role	🗑
Steve Jobs	steve@jobs.com	MDOT	User	Change Role	🗑
Bill Gates	bill@gates.com	MDOT	User	Change Role	🗑

Upon clicking “Change Role”, a modal will appear allowing the administrator to select the role they wish to assign to the user, and submit the change.



Users

Email Address | Invite to Organization

Name					
Hassan Bazzi				ange Role	🗑
Drake Svoboda				ange Role	🗑
Safayeth Khan				ange Role	🗑
Steve Jobs	steve@jobs.com	MDOT	User	Change Role	🗑
Bill Gates	bill@gates.com	MDOT	User	Change Role	🗑

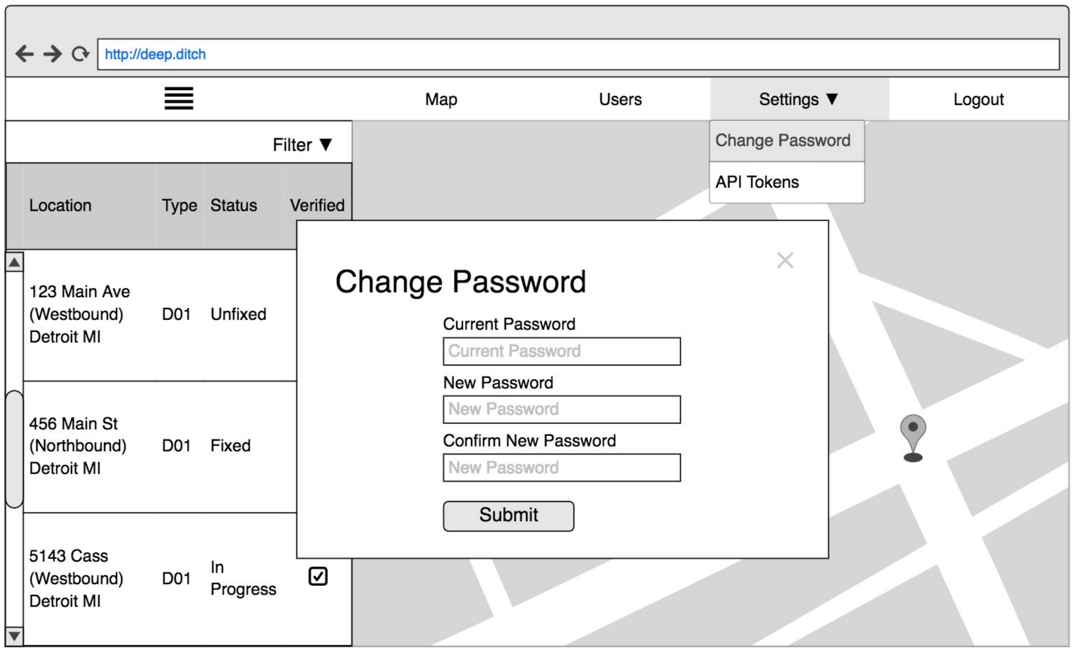
Change Role

Change role for user: Drake Svoboda (drake@svoboda.com)

Organization Admin ▼ | Submit

3.1.2.3 Change Password

All users will be able to change their own password.



Location	Type	Status	Verified
123 Main Ave (Westbound) Detroit MI	D01	Unfixed	
456 Main St (Northbound) Detroit MI	D01	Fixed	
5143 Cass (Westbound) Detroit MI	D01	In Progress	✓

Change Password

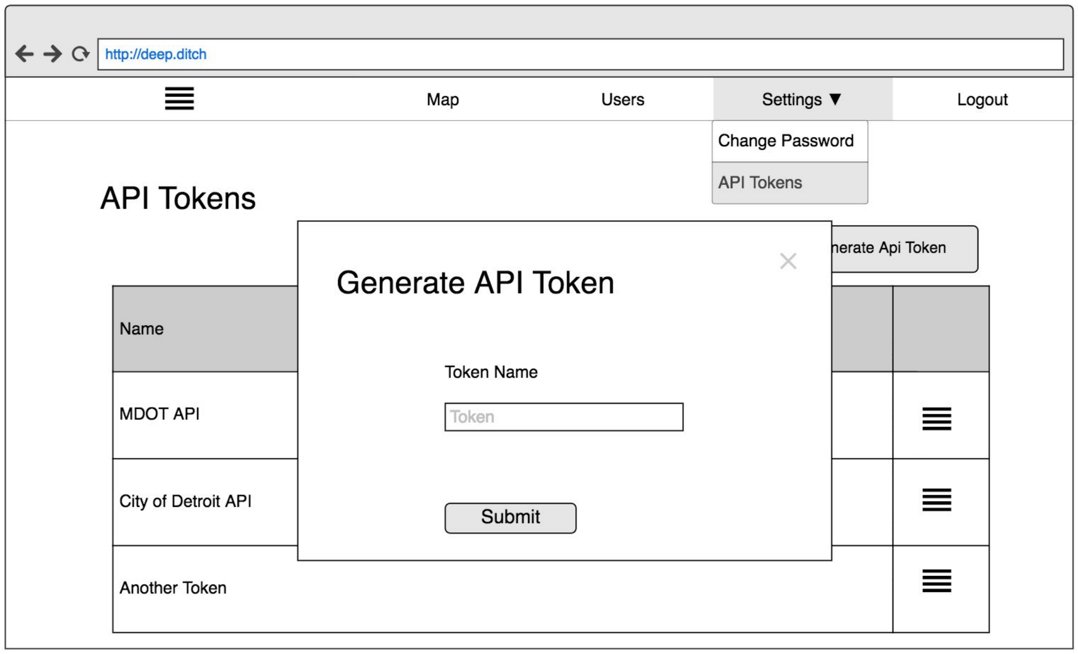
Current Password: Current Password

New Password: New Password

Confirm New Password: New Password

3.1.2.4 API Tokens

An organization admin will be able to generate API tokens.

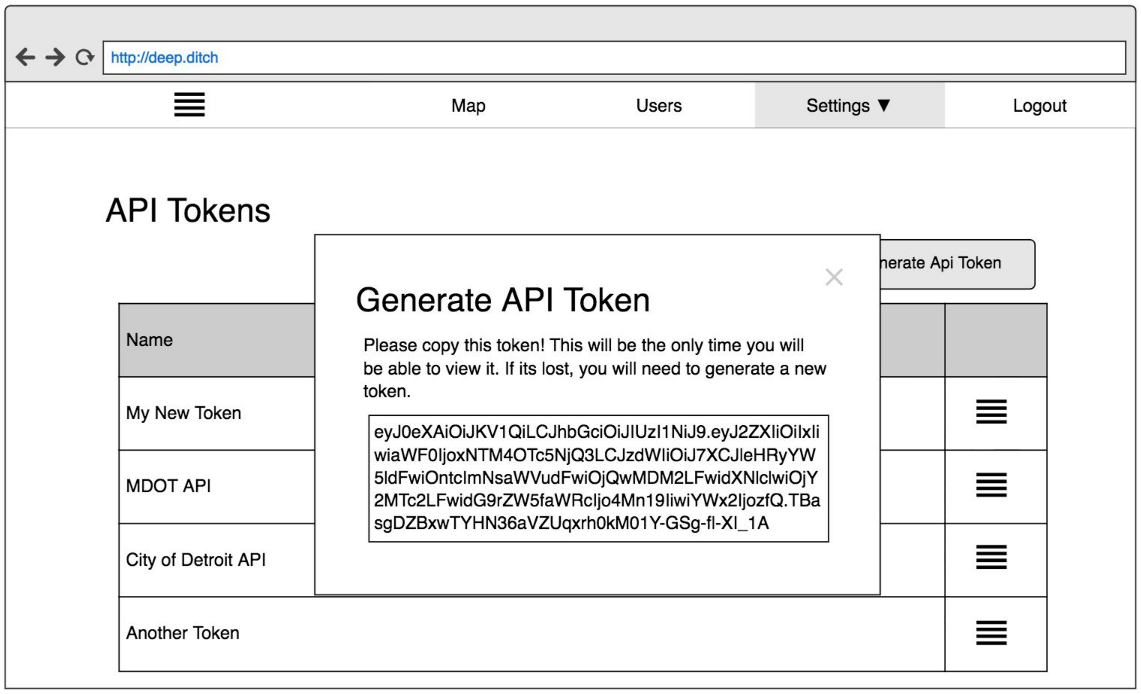


API Tokens

Name
MDOT API
City of Detroit API
Another Token

Generate API Token

Token Name: Token



Generate API Token

Please copy this token! This will be the only time you will be able to view it. If its lost, you will need to generate a new token.

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ2ZXIiOiIxIi
wiaWF0IjoxNTM4OTc5NjQ3LCJzdWIiOiJ7XCJleHRyYW
5ldFwiOntcImNsaWVudFwiOjQwMDM2LFwidXNlclwiOjY
2MTc2LFwidG9rZW5faWRcIjo4Mn19IiwiYWx2IjozfQ.TBa
sgDZBxwTYHN36aVZUqxrh0kM01Y-GSg-fI-XI_1A
```

Name	
My New Token	
MDOT API	
City of Detroit API	
Another Token	

3.1.3 Hardware Interface Requirements

This project will rely on the mobile devices camera, locations services, and internet connection. Any user who wishes to access the web application should have a computer/device with an internet connection and a modern web browser (see section 2.4.1).

3.1.4 Software Interface Requirements

The web application will expose a REST API for retrieving and manipulating data. This will allow for data gathered by our mobile application to be integrated into other software systems.

3.2 Web Application Functional Requirements

This section describes specific features of the system’s web application.

3.2.1 Login

Requirement: A user will be able to login.

Upon visiting the web application—if the user has not yet logged in—the user will be able to complete login form containing inputs for an email address and password. On a successful login, the user will be directed to a page where he or she may view reported road damage. The user will be notified of a failed login attempt.

3.2.2 Register

Requirement: A user will be able to register to use the application.

The user will be able to complete a web form containing inputs for name, email, and password to register. The user’s password will be hashed with salt and stored. Upon successful registration, the user will be redirected to the login screen and an email will be sent to the user’s email address with instructions for verifying his or her account; the user will be notified of this email.

3.2.3 Update Password

Requirement: A user will be able to update their password.

A user will be allowed to change their password after logging in. The user will submit a form with inputs for the old password and the new password. The old password will be verified against the stored password. The new password will be hashed with salt and stored. The old password will be retained in the datastore. The user will be notified that his or her password has been updated and a confirmation email will be delivered to the user’s email address.

3.2.4 Recover a Lost Password

Requirement: A user will be able to recover their account if they forget their password.

If the user does not remember their password, they will be able to request to update their password. The user will provide the email address associated with his or her account and an email will be delivered with instructions for updating the user’s password. The user will then provide the new password they wish to use.

3.2.5 Register an Organization

Requirement: A user will be able to create an organization.

From the registration page, a user may select to simultaneously register an organization. The user will enter the organization’s name, and once registered, the user will be an administrator of the new organization. If a user has already registered, they will be able to create a new organization of which they will be the organization’s administrator.

3.2.6 Send an Email Invitation to Join an Organization

Requirement: An organization’s administrator will be able to invite a user by email to join the organization.

The user must be logged in to perform this action.

An organization’s administrator will be able to deliver invitations to join the organization to any valid email address. If an account for the provided email does not already exist, an email will be delivered to the address with a specialized link to register (3.2.2). If an account with the provided email does exists, an email will be sent to the user with a specialized link to join the organization without registering.

3.2.7 View a List of Users That Belong to the Organization

Requirement: An organization’s administrator will be able to view the list of users who belong to the organization.

The user must be logged in to perform this action.

From this view, the organization's administrator will be able to remove a user from the organization (3.2.8) or appoint a user as an administrator (3.2.9).

3.2.8 Remove a User From an Organization

Requirement: An organization's administrator will be able to remove a user that belongs to the organization from the organization.

The user must be logged in to perform this action.

An organization's administrator will be able to press a button from the user list (3.2.7). This button will remove the user from the organization.

3.2.9 Promote a User as an Organization Administrator

Requirement: An organization's administrator will be able to promote another user to also be an organization administrator.

The user must be logged in to perform this action.

An organization's administrator will be able to press a button from the user list (3.2.7). This button will assign the user to the administrator role. Both users will be notified of a successful promotion.

3.2.10 Demote an Organization Administrator to General User

Requirement: An organization's administrator will be able to demote another administrator to a general user.

The user must be logged in to perform this action.

An organization's administrator will be able to press a button from the user list (3.2.7). This button will assign the user to the general user role. Both users will be notified of a successful demotion.

3.2.11 Create an API Token

Requirement: An organization's administrator will be able to generate persistent API tokens.

The user must be logged in to perform this action.

An organization's administrator will be able to press a button within the web application to generate a unique and persistent API token. This key can be used to access restricted API endpoints. The user will be given the option of naming the API token for easy identification. The user will be returned an expirable API token and password.

3.2.12 Expire an API Token

Requirement: An organization's administrator will be able to expire an API tokens (3.2.11).

The user must be logged in to perform this action.

An organization's administrator will be able to press a button to expire any created API token. Once expired, the API token can no longer be used to access restricted API endpoints.

3.2.13 View a List of Reported Damage

Requirement: A user will be able to view a list of reported road damage.

The user must be logged in to perform this action.

A user will be able to view a list of reported road damage instances in a table form with the location, damage type, and an image of the damage.

3.2.14 View a Map of Reported Damage

Requirement: A user will be able to view a map with a graphical indication of road damage locations.

The user must be logged in to perform this action.

3.2.15 View a Single Damage Instance

Requirement: A user will be able to view the details of an individual damage instance.

The user must be logged in to perform this action.

A user will be able to view the image, type, and status of a an individual damage instance by selecting it from the list of damage (3.2.13) or the map (3.2.14).

3.2.16 Verify a Single Damage Instance

Requirement: A user will be able to manually mark a damage instance as *Verified* or *False Positive*.

The user must be logged in to perform this action.

A user will be able to select *Verified* or *False Positive* while viewing a single damage instance (3.2.15). If the user selects *False Positive,* the damage instance will be retained in the data store; however, it will no longer appear in the default view of the list (3.2.13) or map (3.2.14)

3.2.17 Label a Single Damage Instance's Status

Requirement: A user will be able to mark a damage instance's status as *Pending Repair*, *Repair in progress*, *Repaired*, or *Will not Repair.*

The user must be logged in to perform this action.

A user will be able to select *Pending Repair*, *Repair in progress*, *Repaired*, or *Will not Repair* while viewing a single damage instance (3.2.15).

3.2.18 Filter Damage Instances by Location

Requirement: A user will be able to filter reported damage by location.

The user must be logged in to perform this action.

While viewing the list of damage (3.2.13) or the map (3.2.14), a user will be able to define a specific area and view road damage reported only in that area.

3.2.19 Filter Damage Instances by Type

Requirement: A user will be able to filter reported damage by type.

The user must be logged in to perform this action.

While viewing the list of damage (3.2.13) or the map (3.2.14), a user will be able to specify specific types of road damage and only view road damage classified as the specified types.

3.2.20 Filter Damage Instances by Status Label

Requirement: A user will be able to filter reported damage by status label.

The user must be logged in to perform this action.

While viewing the list of damage (3.2.13) or the map (3.2.14), a user will be able to specify a status label (3.2.17) and view only road damage that has been marked as that status.

3.2.21 Filter Damage Instances by Verification Status

Requirement: A user will be able to filter reported damage by *Verified*, *Unverified* or *False Positive*.

The user must be logged in to perform this action.

While viewing the list of damage (3.2.13) or the map (3.2.14), a user will be able to select *Verified*, *Unverified* or *False Positive*. If the user selects *Verified* or *False Positive*, the user will see only damage instances marked with those labels respectively (3.2.16). If the user selects *Unverified,* the user will see only damage instances that are un-marked.

3.2.22 Retrieve a List of Damage Instances via API

Requirement: The mobile application or an external software client will be able to retrieve a list of reported damage.

The mobile application must provide a user's authentication token to perform this action. The external software client must provide a valid API token (3.2.11) to perform this action.

A client will be able to perform a request to retrieve a list of damage. The client must provide an API token. The web application will respond with the list of road damage belonging to the organization that created the API token (3.2.11). The client will be able to supply filtering options for location (3.2.18), type (3.2.19), status label (3.2.20), and verification status (3.2.20).

3.2.23 Upload a New Damage Instance via API

Requirement: The mobile application or an external software client will be able to upload a damage instance.

The mobile application must provide a user's authentication token to perform this action. The external software client must provide a valid API token (3.2.11) to perform this action.

A client will be able upload a damage instance by providing a type, location, and optional image. Damage uploaded by the mobile app will always have an image (3.3.5).

3.2.24 Verify a Damage Instance via API

Requirement: The mobile application will be able to upload additional classifications for a reported damage instance.

The mobile application will attempt to capture and classify additional images of a reported road damage instance (3.3.7). These additional images and classifications will be delivered to the web API. The web application will be able to weigh multiple classifications to automatically mark reported road damage instances as *False Positive*.

3.2.25 Continuously Train A Machine Learning Model With Verified Images

Requirement: The web application will continuously improve the machine learning model by training on user verified images (3.2.16).

Once a user has verified the correctness of a reported damage instance, that damage instance could be incorporated into that training set and used to improve the classification performance of the machine learning model.

3.3 Mobile Application Functional Requirements

This section describes specific features of the web application portion of the platform.

3.3.1 Login

Requirement: A user will be able to log in.

If the user is not logged in on the device, opening the application the will reveal a login form containing inputs for email and password. The user will complete the form to log in. The user's email and password will be sent to the web API. The web application will verify that the user exists and that the provided password is correct (3.2.1). The web application will return an authentication token. This token will be stored locally on the device for use with future API requests. On a successful login, the user will be redirected to the camera screen. The user will be notified of an unsuccessful login.

3.3.2 Register

Requirement: A user will be able to register to use the application.

From the login screen, the user will be able to navigate to a registration form containing input for name, email, and password. The user will be able to complete the form to register. The contents of the registration form will be sent to the web API. The web application will register the user (3.2.2). On successful registration, the user will be redirected to the login screen and asked to verify their account (3.2.2). Once verified, the user will be able to login. The user will be notified of an unsuccessful registration.

3.3.3 Automatically Identify and Classify Road Damage

Requirement: The application will be able to identify and classify road damage in images captured by the device's camera.

Images captured by the device's camera will be processed in real time by a machine learning model that will determine if those images contain evidence of road damage. If road damage is present in the image, the road damage will be classified into eight possible types (Table 1).

3.3.4 Record the Location of a Road Damage Instance

Requirement: The application will be able to record the location of identified road damage.

When the machine learning model identifies road damage in an image (3.3.3), the location of the identified road damage will be captured using the mobile devices location services.

3.3.5 Report Road Damage to the Web API

Requirement: The application will be able to report identified road damage to the web API.

The user must be logged in to perform this action.

The image, location, and damage type associated with an identified road damage instance will be delivered to the web API (3.3.3, 3.3.4, 3.2.23). If the device does not have an internet connection—but does have location services—the image, location, and damage type will be stored locally and delivered to the web API once an internet connection is re-established.

3.3.6 View a List of Damage Instances

Requirement: Users will be able to view reported road damage.

The user must be logged in to perform this action.

A user will be able to view a list of reported road damage. This road damage will be access from the web applications API (3.2.22).

3.3.7 Verify A Road Damage Instance

Requirement: The application will attempt to automatically verify a reported road damage instance using the machine learning model.

The user must be logged in to perform this action.

If the device is nearing or passing over a previously reported road damage instance, the application will attempt to capture an additional image of that road damage instance. That image will be processed by the machine learning model and a positive or negative result will be reported to the web application (3.2.24).

3.4 Web Application Non-Functional Requirements

These non-functional requirements are regarding the environment in which the web application will be housed.

3.4.1 Web Page Size

Requirement: Any web page delivered by the application will not exceed 10 mb for the initial load of the page.

Additional data may be loaded as required.

3.4.2 Throughput

Requirement: The system will be able to handle around 200 API requests per minute.

We will assume 50 concurrent mobile users at a time. In this case the system shall handle around 200 API requests per minute.

3.4.3 API Query Times

Requirement: The system will achieve an average API response time of 200 ms.

3.4.4 Storage

Requirement: The system will be able to store at least one 300px by 300px image with an average size of 20 kb per road damage entry in the database.

3.4.5 Locations Of Operations

To reduce latency, the server environment will be geographically located within the same region in which the mobile application will be used. In the case of Michigan, the server environment must be located in the Midwest.

3.4.6 Backups and Restoration

Requirement: Backups of the database will be taken every 12 hours to an external server.

The amount of time it will take to restore the database from a backup will depend on the amount of damage instances in the database, however, database maintenance should be performed routinely to optimize the database tables.

Requirement: Backups of the images directory will be taken every 24 hours to an external server.

Each image will be named uniquely, so partial backups are acceptable (only backup images that are not already existent on the destination backup server.

3.4.7 Availability

Requirement: The server environment should maintain 99.9% uptime in terms of network and power availability.

3.5 Mobile Application Non-Functional Requirements

3.5.1 Compatibility

Requirement: The application will support devices running iOS 11 or greater.

3.5.2 Network Conditions

Requirement: The application will gracefully handle a dropped or unavailable internet connection by transitioning to an 'offline' mode.

While offline, data will continue to be gathered and will be delivered once an internet connection is re-established (3.3.5). It is assumed that the device will have available location services while offline.

3.5.3 Data Upload

Requirement: Data upload will not exceed an average of 4 images per minute.

This equates to roughly 80 kb per minute at an assumed 20 kb per 300px by 300px jpg image at an assumed 20 kb per jpg image.

3.5.4 Application Download Size

Requirement: The initial mobile app download will not exceed 65 mb.

3.5.5 Storage

Requirement: The application will never store more than 40 mb worth of data at any given time.

In offline mode, the app will temporarily store images until an internet connection is established (3.3.5). 40 mb of data equates to 2000 300px by 300px jpg images at an assumed 20 kb per image. These 40 mb are in addition to the initial mobile app download (3.5.4) i.e. the total storage footprint of the application will never exceed 105 mb.

3.5.6 Machine Learning Model Updates

Requirement: The machine learning model will be able to be updated without a full application update.

The machine learning model must be able to be updated over an internet connection when an updated model is available.

3.6 Machine Learning Model Non-Functional Requirements

3.6.1 Classification Performance

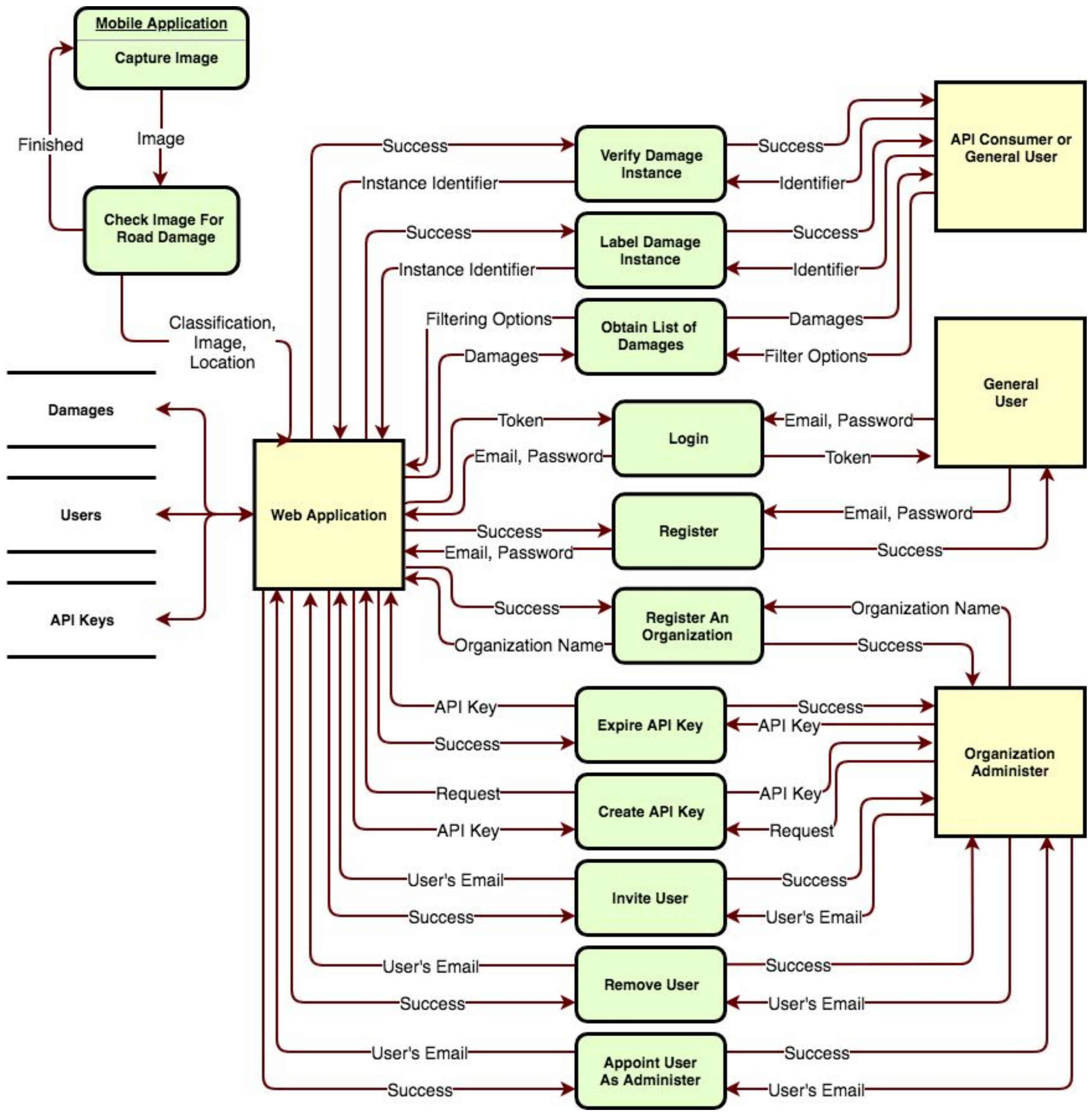
Requirement: The model will achieve an average F-measure of .6 across each of the eight classes (Table 1).

3.6.2 Computational Performance

Requirement: The model will be able to process images at a rate of 2 frames per second on an Apple A10 or greater.

A. Appendices

A.1 Complete Data Flow Diagram


Mobile Application
Capture Image
Finished
Image
Check Image For Road Damage
Classification, Image, Location
Damages
Users
API Keys
Web Application
Verify Damage Instance
Label Damage Instance
Obtain List of Damages
Login
Register
Register An Organization
Expire API Key
Create API Key
Invite User
Remove User
Appoint User As Administer
API Consumer or General User
General User
Organization Administer
Success
Instance Identifier
Identifier
Filtering Options
Damages
Filter Options
Token
Email, Password
Organization Name
API Key
Request
User's Email