Student Lap Tracker

Final Report

Patrick Oyarzun, Joe DeHart
Vaishali Patel, Duke Meche, Robert Knott

Executive Summary

The following document provides complete details about the Student Lap Tracker Project, an application designed to aid teachers in keeping track of their student's laps around a running track. The application consists of a Google App Engine application and a web interface. The application is designed to run indefinitely on google's infrastructure with no cost to the user.

Table of Contents

List of Figures

List Of Tables

1. Introduction

1.1. Purpose and Scope

This document provides comprehensive information about the Student Lap Tracker. The project was completed over the course of 1 semester. During development, decisions were made about the management, requirements, architecture, design, and testing of the software. Those decisions are explained below in the relevant sections.

1.2. Product Overview (including capabilities, scenarios for using the product, etc.)

The product eases lap tracking for a physical education teacher. Currently, as students walk around an outdoor track, the teacher uses an Excel spreadsheet to track each student's total laps.

1.3. Structure of the Document

This document describes the decisions made during project management, requirement generation, architecture design, and test planning in that order. Each section is further divided into relevant parts as noted in the table of contents.

1.4. Terms, Acronyms, and Abbreviations

Herein, the terms ‘software’, ‘product’, ‘program’, and ‘application’ refer to the set of code, documentation, and associated assets required to run the Student Lap Tracker system.
‘Team’ refers to the five members listed in section 2.1.
‘Teacher’, and ‘user’ refers to any person who uses the software.

2. Project Management Plan

2.1. Project Organization

Production Team

Organization and roles of the team members are as follows:

- Patrick Oyarzun - Project Leader, XLSX parsing / data model
- Joseph DeHart - Backend Lap Tracker Code
- Robert Knott - User Interface Implementation / Design
- Vaishali Patel - User Interface Implementation / Design
- Duke Meche - Backend Milestone Management

Rational

Each team member was assigned his/her role based on their existing skillset. The goal in assigning roles is to choose a task for each team member that allows him/her to exercise existing knowledge, while also providing an opportunity to learn. Patrick Oyarzun was chosen as project leader due to his 3 years of experience as Lead Developer at Toast Mobile. He was assigned the task of importing and exporting spreadsheets because it is a complex task, and he uses Google App Engine nearly every day at work, so most other tasks would not yield much of an opportunity to learn. The rest of the team has very similar skill sets and there is less of a justification for their assignments. Joseph DeHart was initially very interested in implementing authentication, but, as detailed later, our authentication strategy is outsourced to Google, so he has shifted roles to aid Duke Meche in the backend implementation of some business logic. Vaishali Patel and Robert Knott had no previous experience with Javascript, and so were tasked with implementing the core ui logic.

Clients

- Devin Olivier
- Eric Credeur

2.2. Lifecycle Model Used

Extreme Programming. Due to our hectic schedules, meeting with each other and with the client is difficult, so other models, like Scrum, are not optimal. By following the extreme programming model, the team will be better able to produce reliable software.
Due to the team's small size and ease of communication with the client, using the extreme programming model would produce better results than more process-focused methods. Also, with the use of GIT version-control, the constant building and rebuilding inherent in extreme programming becomes easier to manage, leading to a more coherent code and less misunderstanding with team members.

2.3. Risk Analysis

- Specification Delays
    - Wait one week from initial contact; if no response is received, then send another email, or talk to Dr. Kumar.
- Absence of Group Member
    - Discuss reason for absence and attempt to find a solution. If no solution is possible, and future absences are expected, distribute work appropriately to make up for the missing person.
- Implementation Failure
    - Explain the situation to the client, apologize, and request an extension.
    - This should mostly be mitigated by requesting constant feedback throughout the development process.
- Group Member Dropping Class
    - Adjust Group Member's roles to make up for the loss

2.4. Hardware and Software Resource Requirements

- Hardware Requirements
    - PC with USB port in order to support the barcode scanner
    - Barcode Scanner which acts a keyboard. During testing, an AGPtek® Portable Wireless USB Barcode Scanner was used.
- Software Requirements
    - Web Browser, testing is carried out in Google Chrome
    - Git
    - Google App Engine SDK
    - Preferred python 2.7 development environment (vim, sublime text, etc.)
- New Technologies Learned
    - Patrick Oyarzun
        - python-excel (xlrd, xlwt, etc)
    - Robert Knott
        - Javascript, HTML, jinja
    - VaishaliPatel
        - Javascript, HTML, jinja
    - Joseph DeHart
        - Google App Engine, Python
    - Duke Meche

        - Google App Engine, Python

2.5. Deliverables and Schedule

1. First Meeting with Techneaux Tech (Sept. 23)
    a. Discuss broad questions of the project, create a set of questions which need investigation.
2. Conference Call with Devin Olivier (Week of Sept. 29)
    a. Ask questions generated during (1). Verify current plan with her and set a time for (3).
3. Presentation of Design Plans with Client (1 week from (2))
    a. Verify design decisions, get feedback
4. Initial Prototype Delivery (2 weeks from (3))
    a. Present first working implementation to client, preferably in person.
    b. Discuss problems in the implementation
    c. Discuss a “wishlist” of idealized features, if any, that the client wishes to have if the project is ahead of schedule, with the understanding that possibly none of them will make it in before launch.
5. Polish the implementation (2 weeks from (3))
    a. Fix problems discovered in (3.b)
    b. Begin work on idealized features only after (4.a)
6. Meeting with Devin Olivier (Following (4))
    a. Show and Tell the ‘mostly’ final project
    b. Discuss final changes needed before launch
7. Make any necessary changes, further refine code
    a. Fix problems discovered during (5)
    b. Spend all extra time testing and validating the system
8. Final Delivery (~Nov. 2)
    a. Demonstrate final product features to client, in person
    b. Discuss how well the project goals have been met
    c. Reflect on lessons learned

2.6. Monitoring, Reporting, and Controlling Mechanisms

Monitoring:

- Software Development plan
    - Due: November 27, 2014
- Integration Management Plan
    - Ensure that the various elements of the project are properly coordinated
    - Due: October 9, 2014
- Scope Management
    - Ensuring that the project includes all of the work required and only the work required
    - Due:October 9, 2014
- Quality Management
    - Ensuring that the project will satisfy the needs for which it was undertaken.
    - Due: November 27, 2014
- Human Resource Management

    - Ensuring that the project makes the most effective use of the people involved with the project.
    - Due: November 27, 2014
- Communications Management
    - Ensuring timely and appropriate generation, collection, dissemination, storage, and disposition of project information.
    - Due: November 27, 2014
- Procurement Management
    - Acquire goods and services from outside organization.
    - Due: October 9, 2014
- Risk Management
    - Identify, analyze and respond to project risk.
    - Due: October 9, 2014
- Cost Management
    - Ensuring that the project is completed within the required budget.
    - Due: October 9, 2014
- Time Management
    - Ensuring that the project completes in a timely fashion.
    - Due: November 27, 2014
- Progress reports
    - Due: Bi-weekly progress reports sent to clients

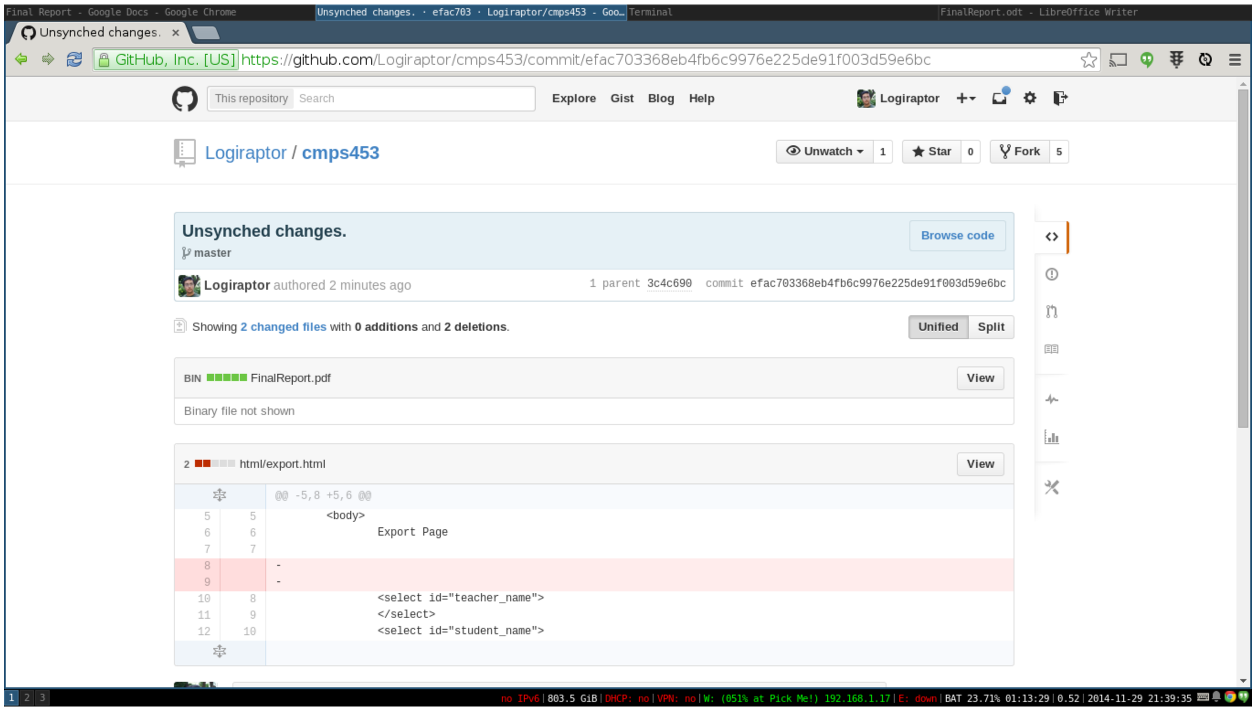
Project Monitoring and Control Mechanisms:

- Communication methods
    - Documentation
        - Strong documentation of all code is required to ensure maximum compatibility and readability
    - Meetings:
        - Regular meetings will be scheduled and attendance is expected.
    - Electronic media
        - Trello will be used for task management and progress control, as well as passive communication
        - Facebook messenger will be used for immediate communication on a team level
- Version Control
    - Git will be used for version control with a remote repository located on GitHub
- Progress Control
    - Tasks will be assigned using Trello
        - All team members will be required to post regular comments on their task cards to make their progress available to the entire team
        - When a task is completed the assigned team member will move the task to the Done “list”. It can then be reviewed and tested by management to ensure compatibility

2.7. Professional Standards

- Team members are expected to attend as many meetings as possible, but it is understood that likely no individual member will be able to attend all meetings.
    - If a member will not attend a meeting, he/she is expected to notify the group either at scheduling time, or at least 24 hours ahead of the meeting date.
- Team members are expected to comply with the university's policy on academic dishonesty. There is absolutely zero tolerance for plagiarism or any other violations.
- Team members are expected to meet deadlines set by the team and notify the other members if he/she will not meet the deadline so the work can be redistributed.
- Team members are expected to meet a certain level of quality in their work. It is understood that many of the tasks assigned are new skills that have never been used before, but a reasonable attempt at producing quality output is expected.
- By formalizing the above expectations, it is easier to maintain a clear set of guidelines for how the team interoperates.

2.8. Evidence all the artifacts have been placed under configuration management



https://github.com/Logiraptor/cmps453/commit/efac703368eb4fb6c9976e225de91f003d59e6bc

Logiraptor / cmps453

Unsynched changes.
master
Logiraptor authored 2 minutes ago   1 parent 3c4c690   commit efac703368eb4fb6c9976e225de91f003d59e6bc

Showing 2 changed files with 0 additions and 2 deletions.

BIN 🟩🟩🟩🟩🟩 FinalReport.pdf
Binary file not shown

2 🟥🟥 html/export.html

```
@@ -5,8 +5,6 @@
        <body>
                Export Page

-
-
                <select id="teacher_name">
                </select>
                <select id="student_name">
```

```
1 2 3                    no IPv6 | 803.5 GiB | DHCP: no | VPN: no | W: (051% at Pick Me!) 192.168.1.17 | E: down | BAT 23.71% 01:13:29 | 0.52 | 2014-11-29 21:39:35
```

Illustration 1: Configuration Management Proof

2.9. Impact of the project on individuals and organizations

Currently, the product is only meant to be used by a single individual. This limits the scope of it's impact to students at the school where it will be used. The impact on those students, however, will be a positive one. The product is used to coordinate rewards for students who achieve their fitness goals. Providing incentives like this will encourage them to work harder towards being healthy.

3. Requirement Specifications

3.1. Stakeholders for the system

A. The user (Physical Education Teacher)
B. The students

3.2 Use Cases

Export Data

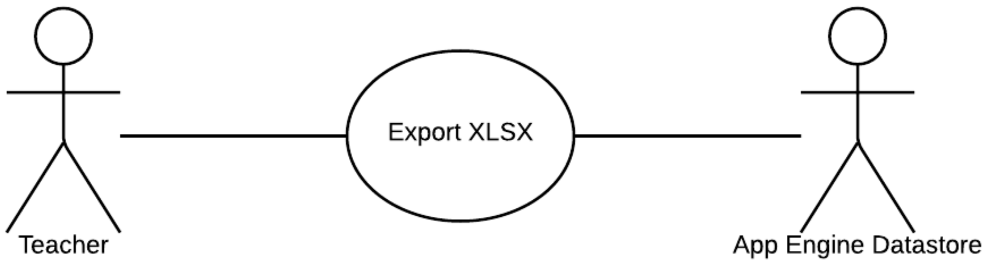


Illustration 2: Export Use Case

The user is able to export the data collected by the application in the form of a csv file. The purpose of this is to allow the user to use spreadsheet software to further process the data.
Participating Actors:
Teacher
Entry Conditions:
At the end of the school year, and perhaps periodically before then, the user wishes to view or process the data using shreadsheet software. He/She clicks a button on the application's main page which initiates the export action.
Normal Flow of Events:
- The user selects the export option from the main page of the application
- The application generates a .xlsx file from the local database and saves it to the selected location.

Exit Conditions:
- The user closes the application through the system close button.

Exceptions:
- The device has insufficient space to save the file
  - The user is alerted that the save was unsuccessful and asked to clear some space on disk and try the process again.
- The user does not have permission to access the directory/file chosen for saving
  - The user is alerted that the save was unsuccessful and asked to choose a different location.

Import Spreadsheet

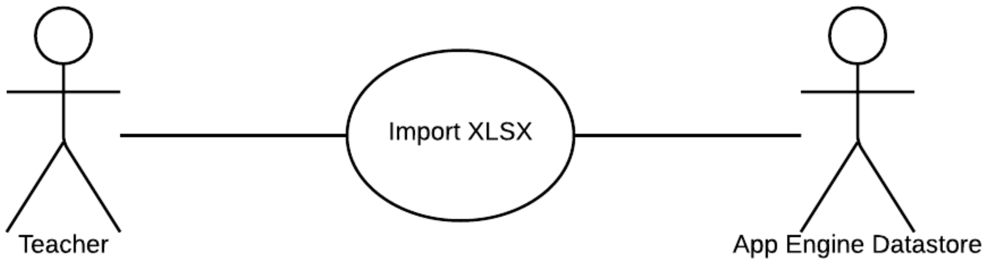


Illustration 3: Import Use Case

The user is able to import existing student data from the previous system (Microsoft Excel) in a xlsx format.

Participating Actors:

Teacher

Entry Conditions:

Upon first receiving the application, the user wishes to import existing data so that current students can be migrated to the new system without losing any data. He/She selects the import option from the application's main page and the import process is started.

Normal Flow of Events:

- The user selects the import option from the main page of the application
- A file chooser dialog prompts the user to choose a location on disk
- The application parses the xlsx file and loads the data into the application's local database.

Exit Conditions:

- The user cancels the process by closing the file chooser dialog, or pressing the cancel button on the file chooser.
- The user closes the application by clicking the system close button.

Exceptions:

- The xlsx file is malformed or corrupted.
    - The user is alerted that the import was unsuccessful and advised to try exporting the data from excel again, or correct invalid data
- The user has insufficient permission to access the selected file
    - The user is alerted that the import was unsuccessful and advised to choose a different file or change the file's permissions.

Track Lap

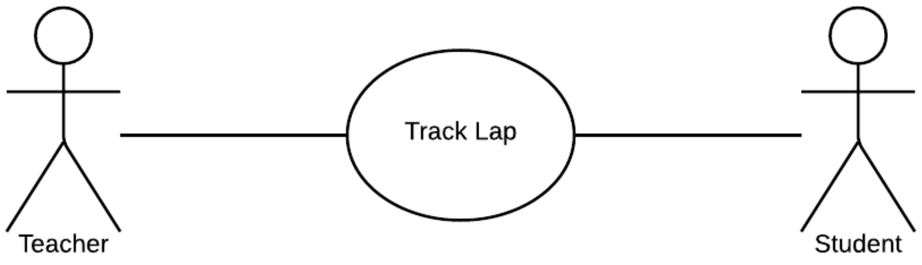


Illustration 4: Track Lap Use Case

The user is able to track students' performance as they complete laps around one of two tracks.
Participating Actors:
Teacher, Student
Entry Conditions:
Each day, the teacher wishes to record student's laps as they are completed. He/She selects the track option from the main screen of the application.
Normal Flow of Events:
- The user selects the track option from the main page of the application
- The user is prompted to select which of the two tracks the students are running on today
- The application enters a waiting state in which it waits for one of several things to happen
    - The user scans a barcode
        - The application adds one lap to the student with a matching id
    - The user manually changes a record
        - In the event that a student forgets his/her id or the scanner is accidentally used twice on the same id, the application provides a method of selecting a student and entering a new value for the lap counter.

Exit Conditions:
- The user cancels the process by closing the application
- The user navigates back to the main application screen

Exceptions:
- A student forgets to bring an id to school
    - See Normal Flow above.
- The user accidentally enters duplicate data
    - See Normal Flow above
- The barcode scanner is unable to read a student's id
    - The user can opt to use the manual entry feature to input the correct data

Special Requirements:
The barcode scanner (HX-US470) must be working and available for use.

Non-Functional Requirements

Product Requirements

- Security Requirements
    - The student data imported via XLSX shall be kept private to prevent unauthorized access.

Organizational Requirements

- Environmental Requirements
    - The product requires an internet connection to sync with App Engine, which allows importing and exporting of xlsx files.
- Developmental Requirements
    - Developing the product may require a similar barcode scanner in order to test functionality.

4. Architecture

4.1. Architectural style(s) used

The application will be implemented using the Model View Controller (MVC) style to coordinate implementation of features without introducing unwanted complexity into the system. The primary concern of the application is managing and maintaining a dataset of students and their respective progress on a running track.

4.2. Architectural model

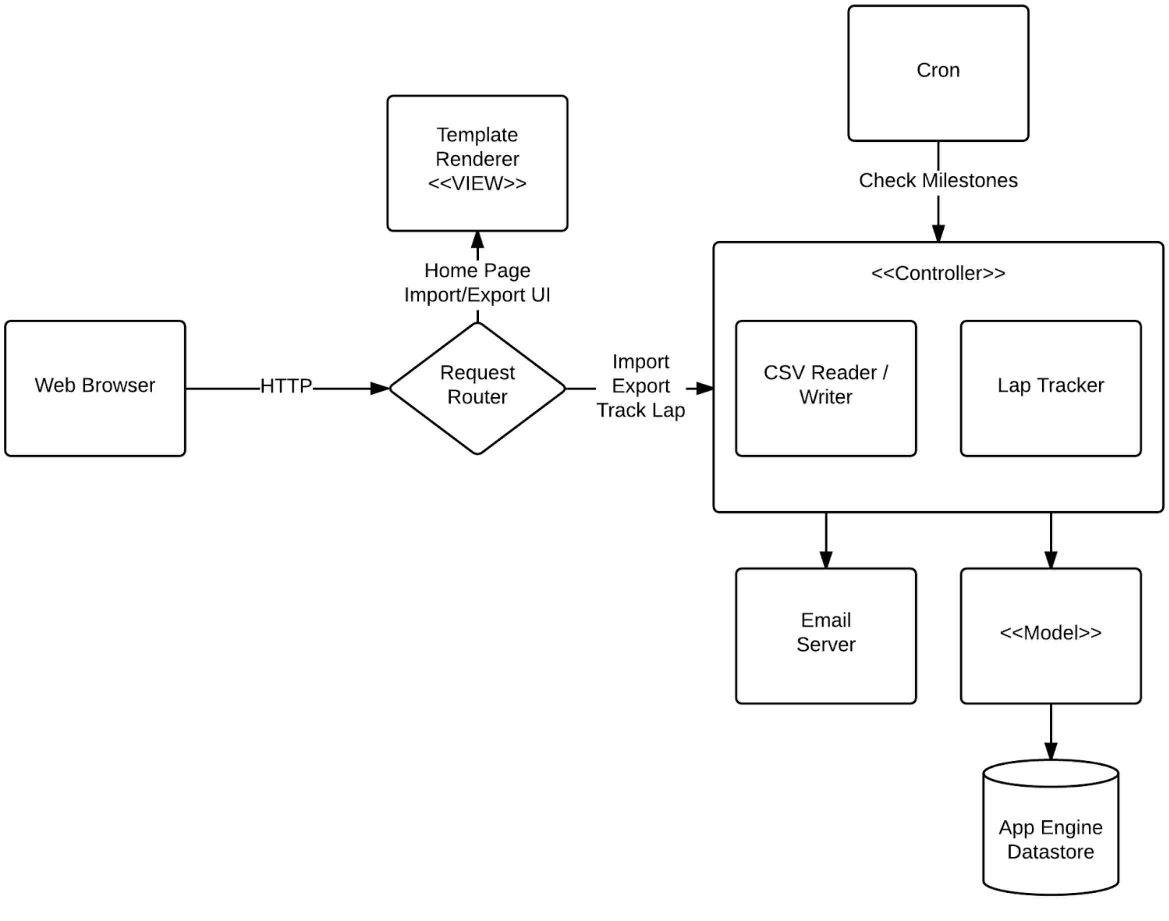


Illustration 5: Architectural Model

4.3. Technology, software, and hardware used

Technology

The server is hosted on Google App Engine under the Python runtime. App Engine is a platform-as-a-service(PAAS) solution designed to allow small teams to develop distributed applications with minimal work. Several components of the application use the services provided as part of the platform. The underlying database is the App Engine datastore, a highly scalable distributed database built upon a distributed storage system called Big Table. Our application will not likely require the scalability features provided by the datastore, but having them available with zero configuration increases the scope of use cases our application can grow to support in the future. The application also makes use of Google's distributed memcache architecture, the email service, and a message queue which allows maintenance tasks to run on a schedule, or outside the context of an HTTP request.

Software and Hardware Requirements

1. The App Engine Python sdk
2. Web browser
3. Developer's preferred environment for developing python applications, e.g. Sublime Text 2.
4. In order to receive email reports, an email client is required.
5. A barcode scanner with the ability to emulate a standard keyboard
6. Web-connected machine with a usb port and support for keyboard use.

Communication

The application server communicates with the datastore through protocol buffer RPC requests. This is handled transparently by the AppEngine sdk. Client server communication is accomplished through a JSON REST interface. The client is implemented in javascript, so a JSON interface is ideal in order to minimize the development time overhead associated with communication.

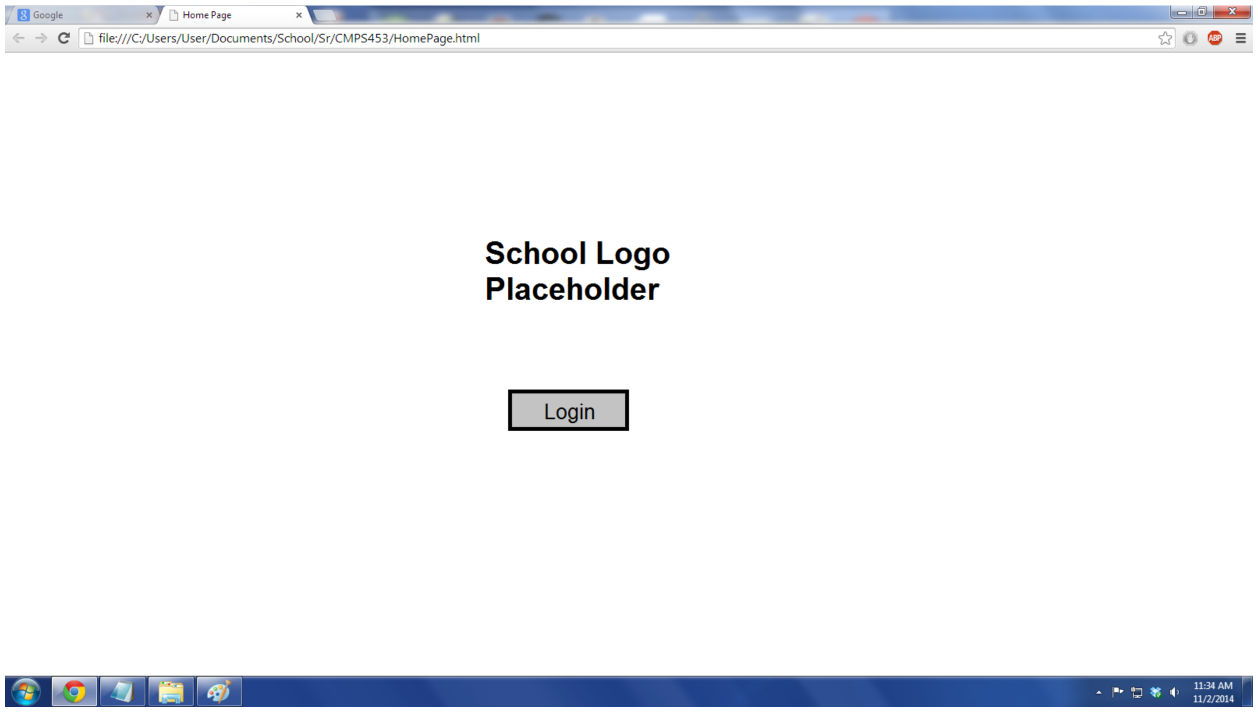
4.4. Rationale for your architectural style and model

The primary user interface is implemented through a web browser, and MVC allows the application logic to remain straightforward in terms of separating concerns and ensuring the application's invariants are maintained. MVC is a natural fit for web applications due to the inherent decoupling between frontend and backend logic that is imposed by the system. Google App Engine was chosen due to the fact that, for small request loads, the service is entirely free. Our focus is on delivering a quality product and using App Engine allows the team to focus on the business logic without being slowed down by system-level concerns which are outside the scope of the project. For example, load balancing, server restarts, and routine server maintenance are all handled seamlessly by Google while providing near-zero downtime for our application.

5. Design

5.1. GUI (Graphical User Interface) design

- **Home Page**: Contains the school logo and information as well as a button linking to the Google Account Sign-In.

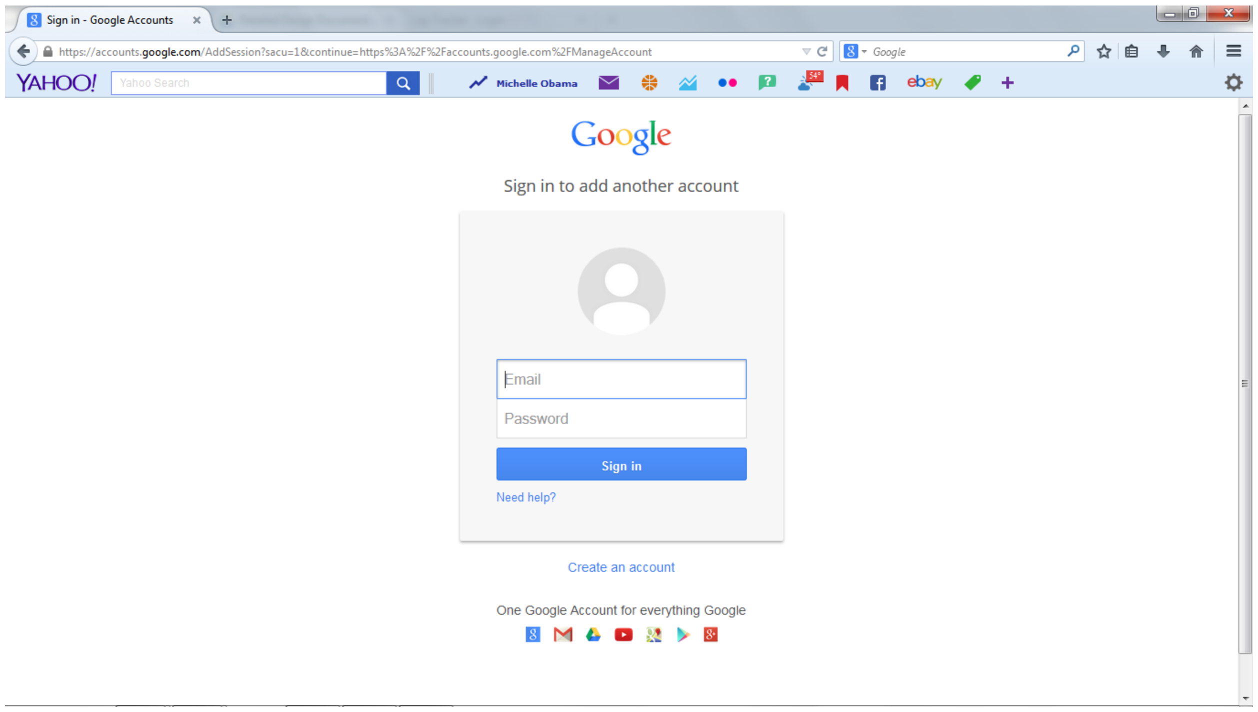

Google
Home Page
file:///C:/Users/User/Documents/School/Sr/CMPS453/HomePage.html

School Logo Placeholder

Login

11:34 AM
11/2/2014

Login Page: Google provides our login page. Only an admin will be able to access all of the pages.


Sign in - Google Accounts
https://accounts.google.com/AddSession?sacu=1&continue=https%3A%2F%2Faccounts.google.com%2FManageAccount

Google

Sign in to add another account

Email

Password

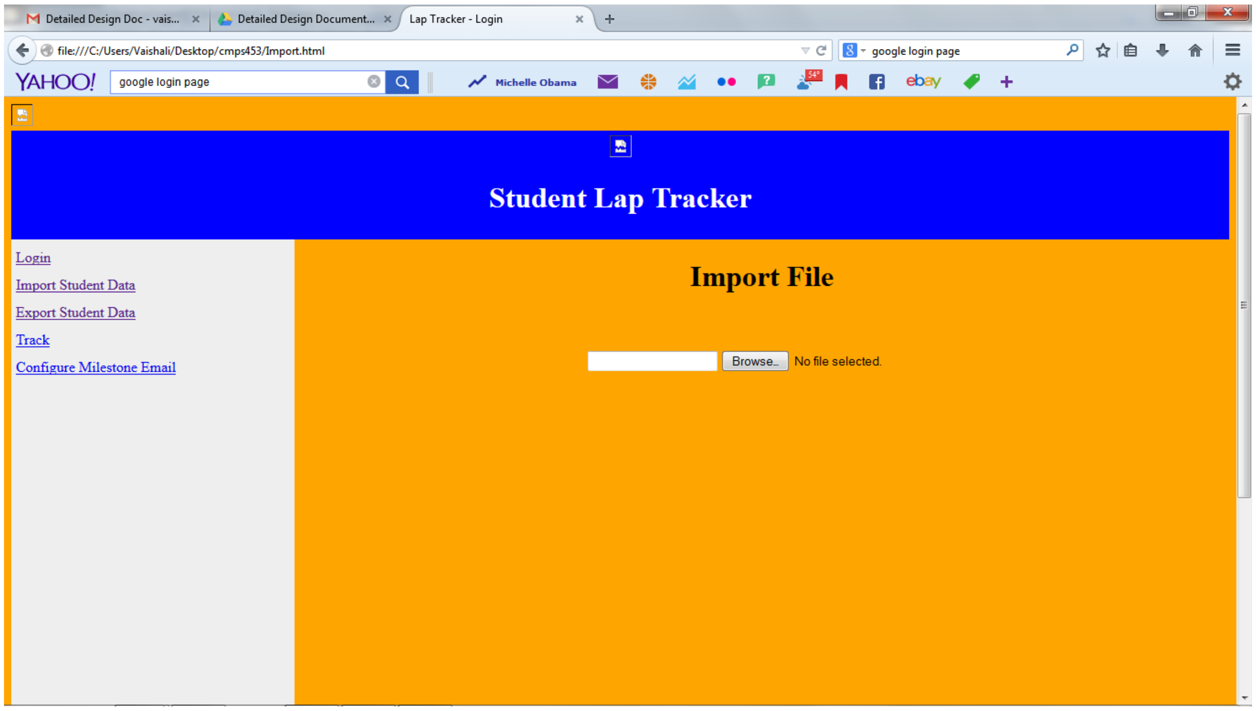
Sign in

Need help?

Create an account

One Google Account for everything Google

Import Page: A user can imports an Excel Spreadsheet file with .xlsx, which contains the students data through the browse button and the data of the selected file will be displayed on the page.


Detailed Design Doc - vais...
Detailed Design Document...
Lap Tracker - Login
file:///C:/Users/Vaishali/Desktop/cmps453/Import.html
google login page
YAHOO!
Michelle Obama
ebay

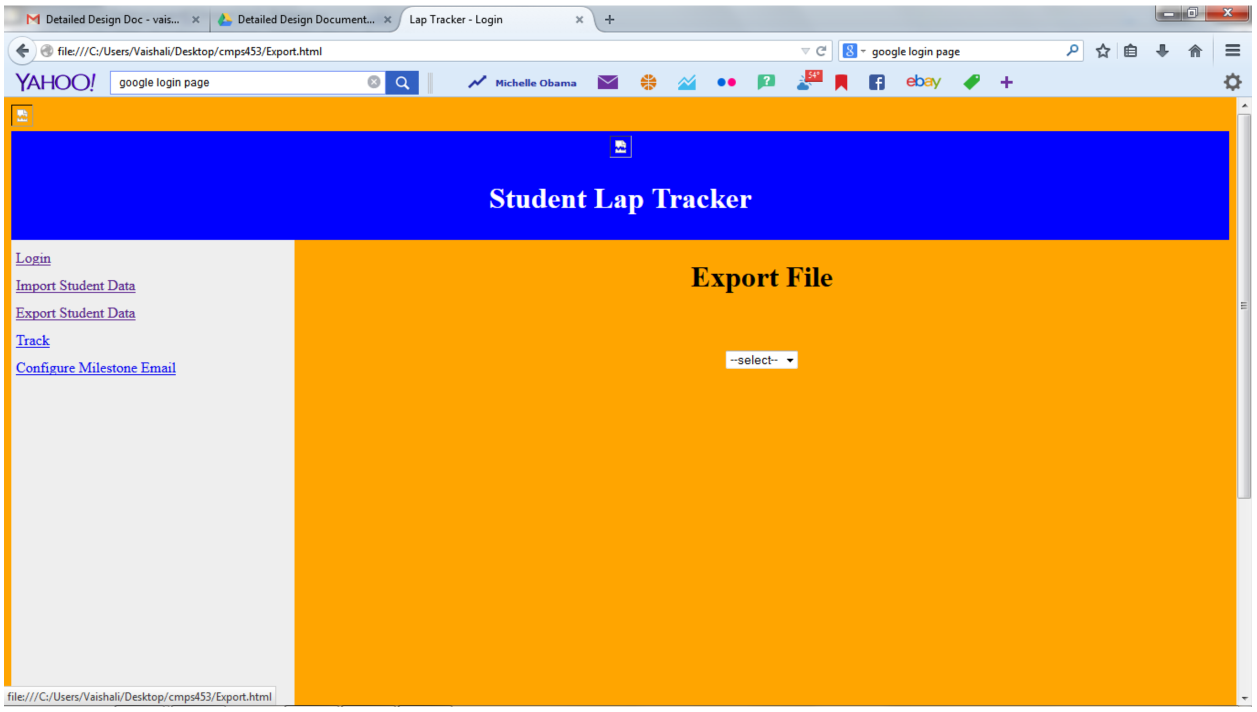
Student Lap Tracker

Login
Import Student Data
Export Student Data
Track
Configure Milestone Email

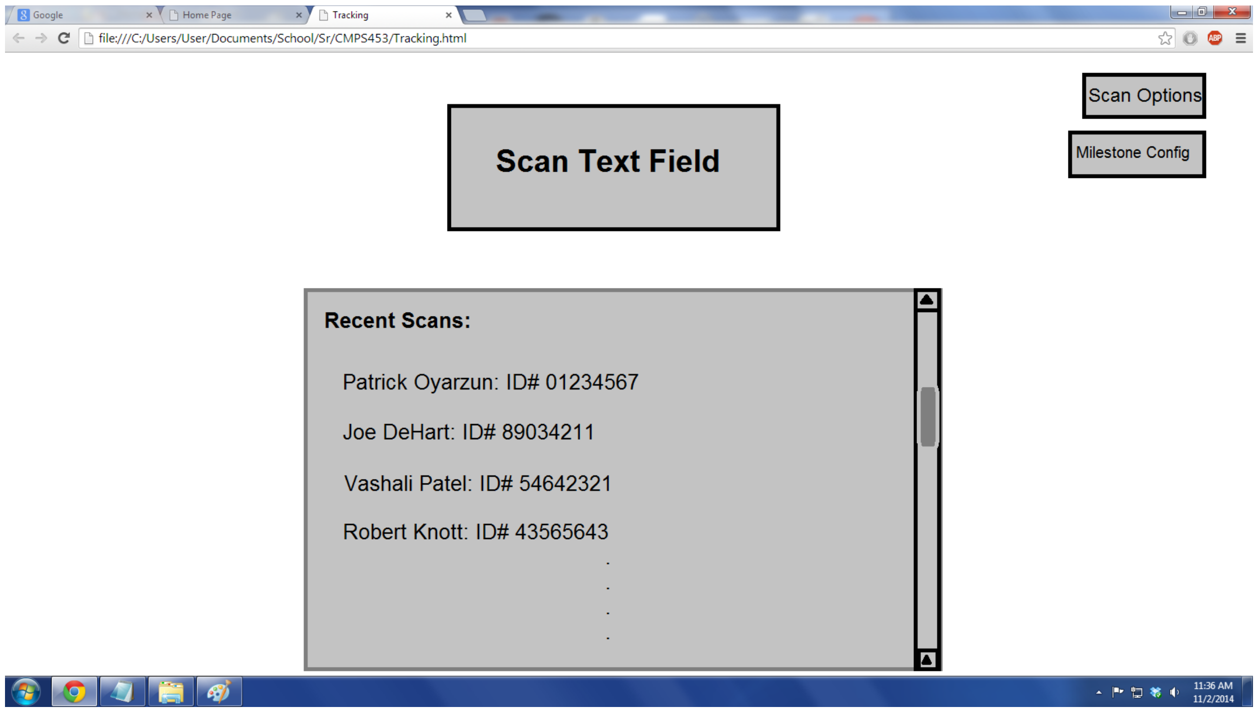
Import File

Browse... No file selected.

Export File: An export page will either display the data of all the students or the data of a particular student according to the selection of a drop down menu.


Detailed Design Doc - vais...
Detailed Design Document...
Lap Tracker - Login
file:///C:/Users/Vaishali/Desktop/cmps453/Export.html
google login page
YAHOO!
google login page
Michelle Obama
Student Lap Tracker
Login
Import Student Data
Export Student Data
Track
Configure Milestone Email
Export File
--select--
file:///C:/Users/Vaishali/Desktop/cmps453/Export.html

Tracking and Scan Page: The user navigates to this page to scan barcodes that will send upload requests to the App Engine Data Storage. There is also a Recent Scans window that will show a record of the most recent scans made. There are also two buttons used for modifications; Scan Options which will allow the user to define which course should be used to increment each ID scan, and Milestone Config, which allows the user to modify their Milestone goals.



Recent Scans:

Patrick Oyarzun: ID# 01234567

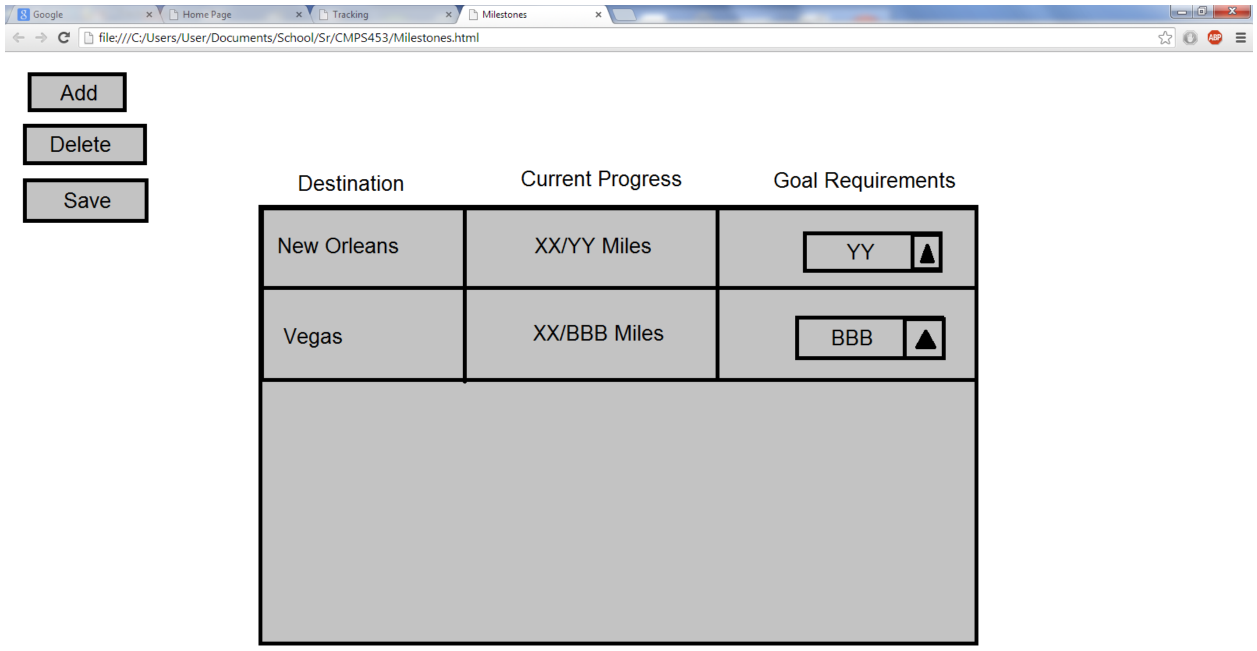
Joe DeHart: ID# 89034211

Vashali Patel: ID# 54642321

Robert Knott: ID# 43565643

.
.
.
.

Milestone Config: This page allows the user to add, delete, or modify any of their currently active milestones. It will send modification requests every time the Save button is pressed, and will automatically send one when the Add or Delete functions perform successfully. Should a Milestone be reached, it will automatically send a request to server that will e-mail the linked email address about the accomplishment.


Google
Home Page
Tracking
Milestones
file:///C:/Users/User/Documents/School/Sr/CMPS453/Milestones.html
Add
Delete
Save

Destination	Current Progress	Goal Requirements
New Orleans	XX/YY Miles	YY
Vegas	XX/BBB Miles	BBB

5.2. Static model – class diagrams

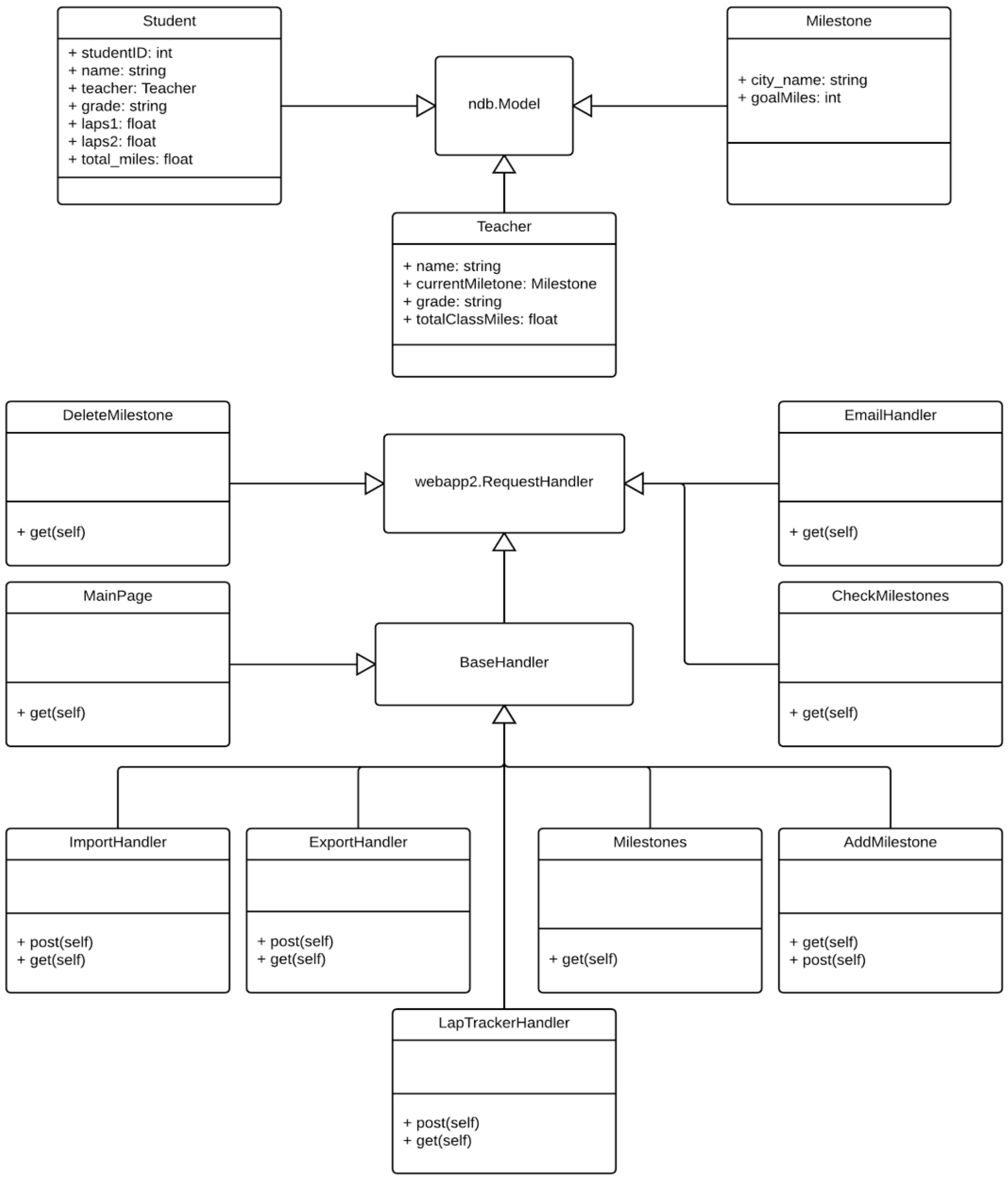


Illustration 6: Class Diagram

Data persistence model:

The database schema is composed of three models. Due to the nature of the appengine datastore, there is no explicit schema enforced by the database. Instead each entity “Kind”, analogous to an RDBMS table, is defined by a normal python class definition which subclasses the ndb.Model class. All operations are then inherited by the subclass automatically. The three models are as follows:

Teacher

ID	Unique Identifier assigned by the appengine infrastructure
Name	Teacher’s name taken from an uploaded excel spreadsheet
Grade	Teacher’s grade taken from an uploaded excel spreadsheet
CurrentMilestone	The Milestone that the Teacher’s class is currently working towards.
TotalClassMiles	The total number of miles walked by the Students in the Teacher’s class.

Table 1. Teacher Entity

Student

StudentID	Unique Identifier assigned by the school
Name	Student name taken from uploaded Excel spreadsheet
Teacher	The student’s teacher, obtained from uploaded data
Grade	The student’s grade, obtained from uploaded data
Laps1	The most recent sum of the student’s laps on track 1, this value is maintained to speed up later calculations involving the total miles a student has walked.
Laps2	The most recent sum of the student’s laps on track 2, this value is maintained to speed up later calculations involving the total miles a student has walked.
Total_Miles	The total number of miles the student has walked, which is Laps1 * 0.1 + Laps2 * 0.25.

Table 2. Student Entity

Milestone

ID	A unique identifier assigned by the appengine infrastructure
City_Name	The name of the milestone, usually named after a city.
GoalMiles	The target number of miles the students must walk to get to the city.

Table 3. Milestone Entity

5.3. Dynamic model – sequence diagrams

- **IMPORT SEQUENCE**

The import flow involves three major steps. First, the data is uploaded to the server through an HTML form in the client implementation in the form of a Microsoft Excel spreadsheet. This data is then passed to a reader library called xlrd, which is part of the python-excel project [1]. The resulting workbook object is used to construct a set of database model instances which represent the uploaded data. Once these instances are constructed, they are inserted into the appengine datastore in a batch put operation which allows all the writes to occur in parallel on the distributed infrastructure. The user is then redirected to a summary page which gives an overview of the uploaded data so that the user can verify the import process was successful. In certain cases an import may fail due to corrupt data, service unavailability, etc. In any case where the error is detectable by the server code, an informative error message is shown instead of the summary page.

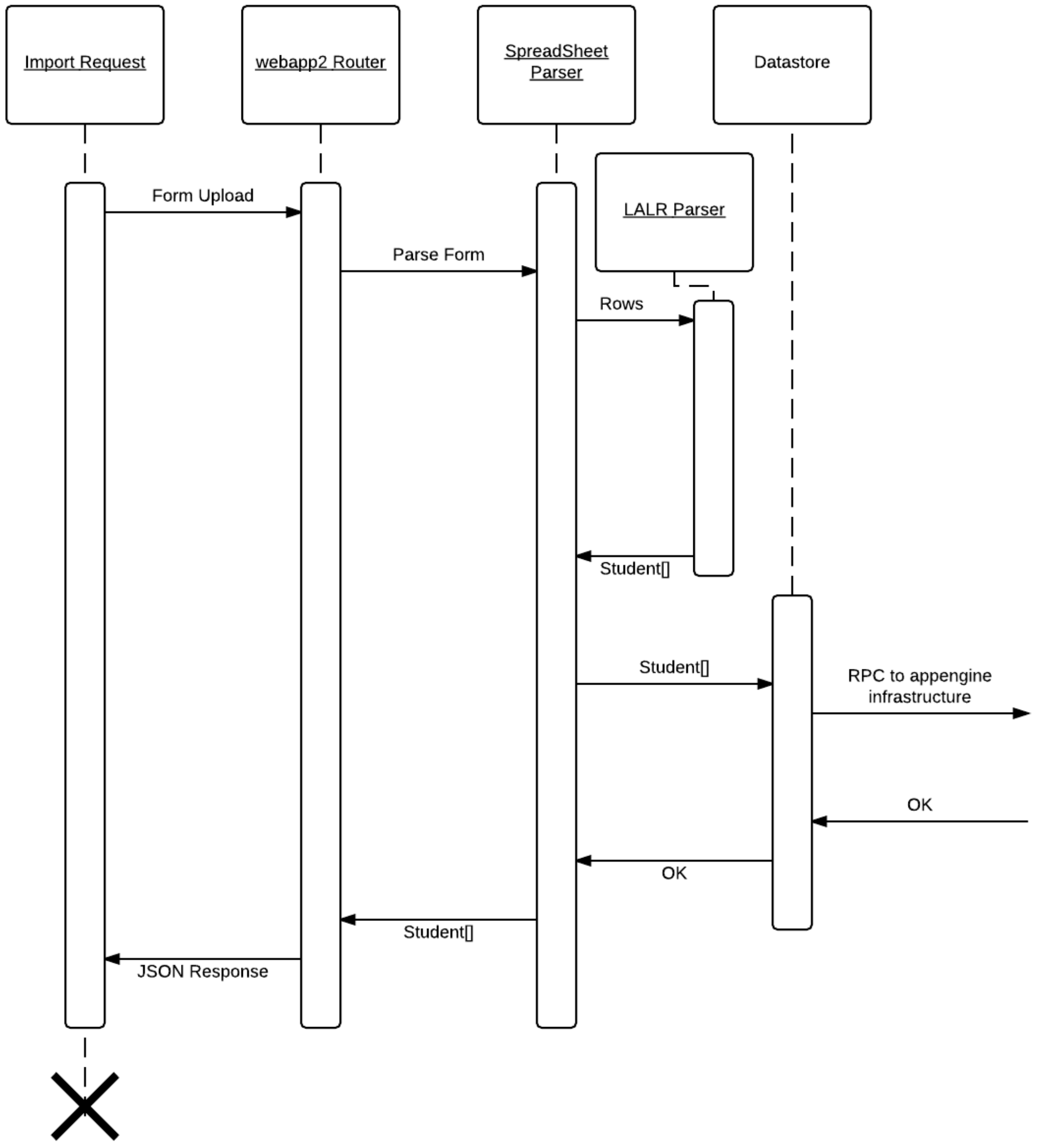


Figure #. Import Sequence

- **EXPORT SEQUENCE**

The export sequence is simpler than the import flow since the data is in a predictable, structured format inside the appengine datastore. Therefore there are two main stages. First, a series of database queries are executed in order to obtain the relevant data depending on the subset of data the user is interested in exporting. This may be a system-wide backup, or a single student. That data is then used in conjunction with the xlwt library from the python-excel project [1] to construct the workbook object. This object can then be serialized to the http response.

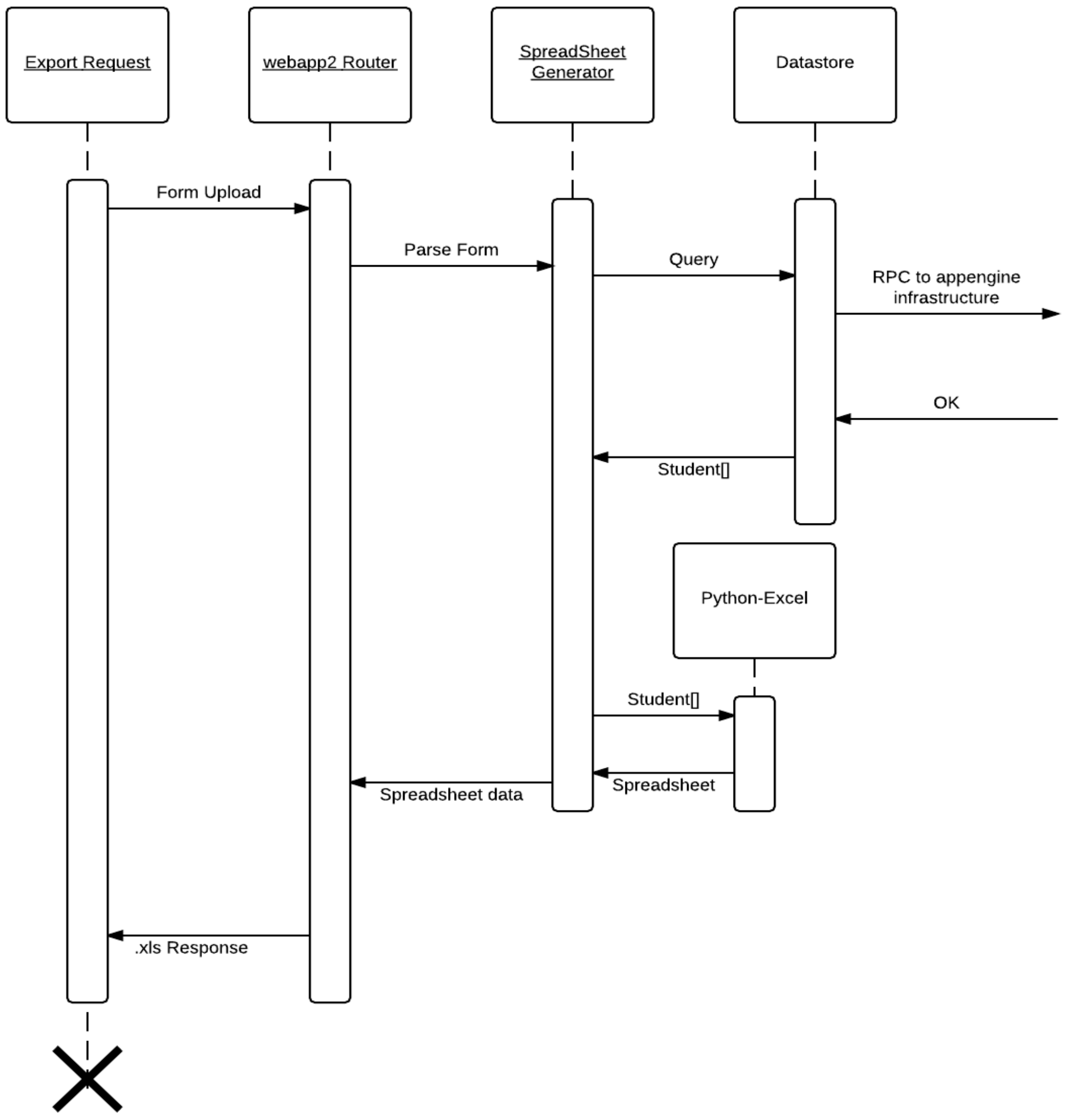


Illustration 7: Export Sequence Diagram

- **AUTHENTICATION SEQUENCE**

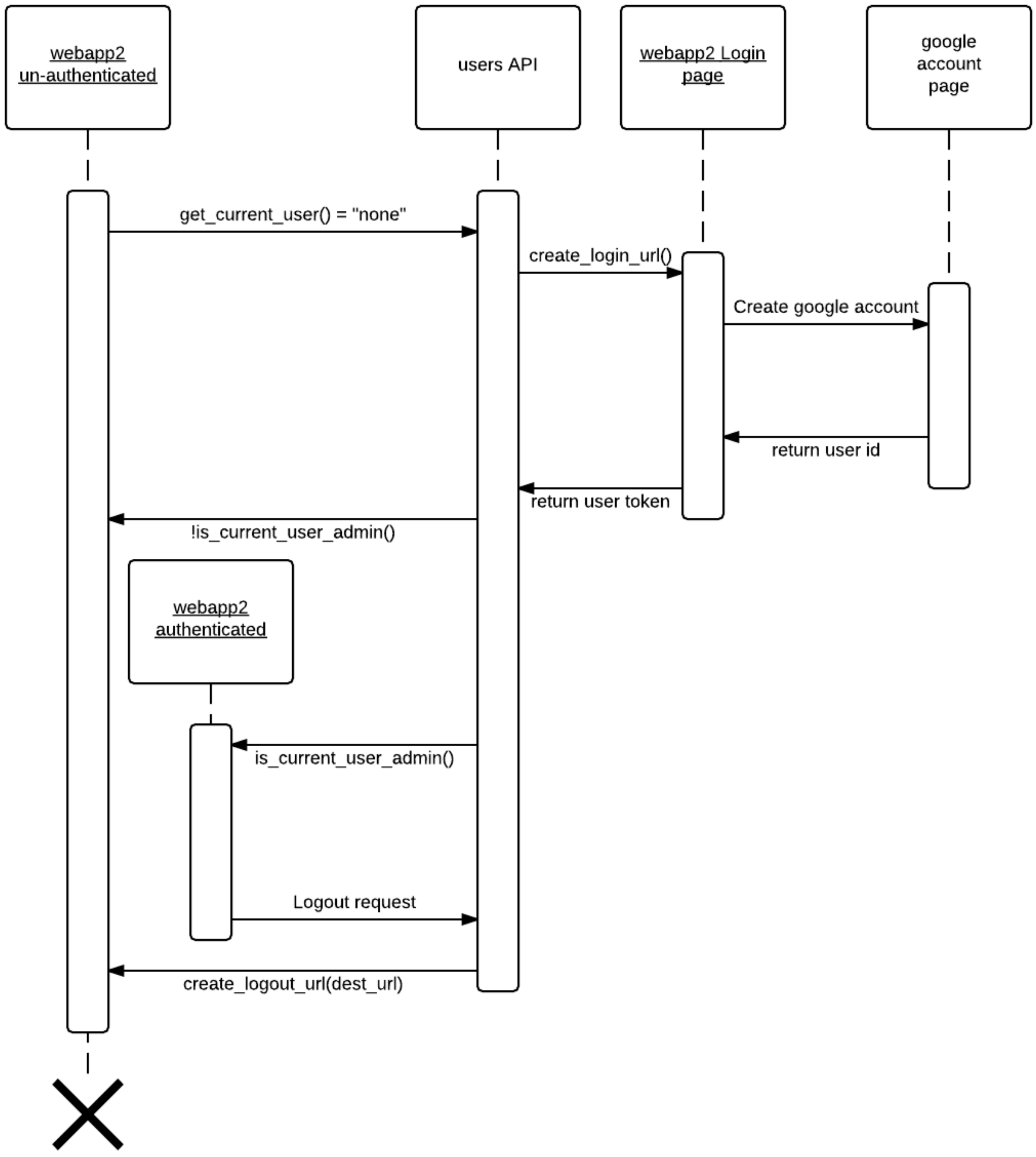


Illustration 8: Auth Sequence Diagram

5.4. Rationale for the detailed design model

A primary goal of this project is to provide a minimum cost solution to the user. The project was given a $0.00 budget to work with, so it is important that any and all costs are reduced wherever feasible. The primary source of cost in many appengine applications is datastore usage. Billing is usage-based, and all applications receive a quota on each available resource under which there is no charge. To be clear, this quota resets every 24 hours, so as long as an application remains below the quota during each day, it will be refilled. For the appengine datastore, this quota is 50,000 read operations, 50,000 write operations, and 50,000 ‘small’ operations [2]. Read and write are self-explanatory. A small operation is an operation that can be executed cheaply by google’s infrastructure. This includes things like fetching an entity directly by its ID. In order to minimize our use of the datastore without losing any data, the application uses a caching strategy which makes effective use of the appengine distributed memcache as well as a few extra properties on the student and lap models to optimize our queries. Each time a lap is entered into the system, a single row is generated in the datastore. Its counted field is marked false to indicate that it has not yet been added to its student’s lap totals. Each day as the milestone cron job is executing, all lap rows which have not yet been counted are collected and their quantities are added to the relevant students. This ensures there will be no contention when adding laps in bulk since each one is completely independent. It also ensures that the application can quickly approximate any student’s total mileage as well as the system mileage without having to perform an expensive sum operation on the lap table.

5.5. Traceability from requirements to detailed design model

The requirements documentation lists three primary use cases, Import, Export, and Track. Above are listed sequence diagrams for import / export. Also listed is a specification of the data model and rationale involved in implementing the lap tracking use case.

6. Test Plan

6.1. Requirements/specifications-based system level test cases

Integration Tests

Authentication

Visit Homepage without Authentication

- Expected Output
    - Since the request handlers are decorated with either @login_required or @admin_required, the user will be redirected to a google account login page.

Visit Homepage with Authentication

- Expected Output
    - A list of links for each of the other pages: import, export, lap tracking, and milestone configuration.

Import

Upload Correct Workbook

- Expected Output
    - After importing an excel spreadsheet, the contents of the file will be displayed on the page, if processed successfully.

Upload Workbook with mismatched teacher names

- Expected Output
    - Displays an error message that the teacher name does not match with the database record.

Upload Workbook with mismatched student names

- Expected Output
    - Displays an error message that the student name does not match with the database record.

Export

Download All Students

- Expected Output
    - Displays the data of all the students, if drop down menu selection is all students.

Download Single Student

- Expected Output:
    - If drop down menu selection is the student id of any particular student, then it will only display the data of a particular student associated with that selected id.

Tracking UI

Scan Barcode

- Expected Output
    - After scanning a student's ID barcode, a success message is displayed and the scan is added into the Recent Scan window.

Undo a repeated scan

- Expected Output
    - If a scan was unnecessary or repeated by accident, the user can remove the scan by clicking the delete button on the scan's success message in the Recent Scan window

Milestones

Display Milestones

- Expected Output
    - A list/table of the milestones that are currently active.

Add Milestone

- Expected Output
    - A new row with the milestone specifications is added to the list/table. A success message appears briefly to let the user know the addition was successful.

Delete Milestone

- Expected Output
    - The specified milestone is removed from the list/table. A success message appears briefly to let the user know the removal was successful.

Modifying Milestone

- Expected Output
    - The milestone's specifications are changed to the user's new requirements. A success message appears briefly to let the user know the change was successful.

Cron Emails

Summary Email of Milestone Progress (Weekly Fridays after school)

- Expected Output
    - This will be called weekly after school and will send an email to the teacher containing a milestone progress report which may look like the following:

    Weekly Milestone Progress Report:

Teacher Name	Laps	Laps to Go Until Next MileStone
Duke Meche	61	39
Vaishali Patel	74	26
...		
Patrick Oyarzun	79	21

Trigger Milestone (Daily weekdays after school)

- Expected Output
  - In administrator console:

    The Google App Engine administrator console provides a way to check the functionality of each cron job that is active. It provides the team with the latest scheduled tasks, when they are scheduled to be executed, and whether or not the job was successful.

/crontask daily milestone check	every day 16:30 (US/Central) 2014/11/07 16:30:00 **on time Success**

- Other:

1. If the milestones have not been reached by any student, nothing is done.
2. If the milestones have been reached, the teacher will be notified via email which teacher has passed the milestones and which class should be rewarded.

Unit Tests for Internal Server Implementation

Import

Parse Correctly formatted Workbook

- Expected Output
  - The datastore contains a Teacher entity for each teacher in the spreadsheet.
  - The datastore contains a Student entity for each student in the spreadsheet.
  - Each student entity has the correct counts for track 1 and track 2.

Parse Workbook with mismatched teacher names

- Expected Output
  - Nothing is stored in the datastore.
  - An error message is returned naming the teacher which was not able to be matched.

Parse Workbook with mismatched student names

- Expected Output
  - Nothing is stored in the datastore.
  - An error message is returned naming the teacher which was not able to be matched.

Export

Download All Students

- Expected Output
    - A single Excel Workbook (.xlsx) is generated and it contains the entire contents of the database in the same format as the original uploaded spreadsheet.

Download Single Student

- Expected Output
    - The student object is returned singularly. This functionality does not generate an Excel Workbook as that would be superfluous for a single row.

Tracking UI

Scan Barcode

- Expected Output
    - The barcode number must match a student which exists in the datastore
        - If not, an error occurs.
    - The referenced student has its lap counter field incremented automatically
    - A record of the scan is added to the datastore

Undo a repeated scan

- Expected Output
    - The id submitted must reference an existing scan entity.
        - If not, an error occurs
    - The scan entity referenced by the id is deleted, and the student which was originally scanned has its lap counter field decremented automatically

Milestones

List Milestones

- Expected Output
    - Returns a list of milestones.
    - If there are no milestones currently in the Datastore, notify the user.

Add Milestone

- Expected Output
    - The new milestone containing a reference name and an integer goal value is added to the milestone table in the Datastore.
        - If any non-integer values or strings are presented as a goal, an error will occur.

Delete Milestone

- Expected Output
    - The milestone chosen by the user is removed from the Datastore.
        - The milestone chosen to be deleted must reference an active milestone, else an error will occur.

Modifying Milestone

- Expected Output
    - Milestones are changed to match new requirements.
        - If the milestone that is to be changed does not reference an active milestone, an error will occur.
        - If the new goal value is a non-integer value, an error will occur.

Cron Emails

Summary Email of Milestone Progress (Weekly Fridays after school)

- Expected Output

        - Email is dispatched to target containing a summary of milestone progress.

Trigger Milestone (Daily weekdays after school)

- Expected Output
    - If the milestones are met, an email is sent to the teacher with the names of the teachers whose classes have passed the milestones.

6.2. Traceability of test cases to use cases

Each of the original three use cases is covered by at least 1 integration test and several unit tests internally. These include Import, Export, and Track. This verifies that the spec is being followed. Additionally, the access control requirements are verified through some additional integration testing.

6.3. Techniques used for test generation

The integration tests are all treated as black-box tests. This is a consequence of a client-server architecture. In order to verify the client side implementation, the tests must not depend on any implementation detail. The client has no access to the internal application state without passing through the MVC architecture on the server.

The unit tests on the other hand must use internal implementation knowledge to verify the pre-conditions and post-conditions of all functions are correct. It is not possible to verify a database table was properly updated without knowing which table is involved in the transaction.

6.4. Assessment of the goodness of the test suite

Test quality is measured along a single axis: coverage. A test with 100% coverage is the ideal test. Coverage is a measure of how thorough a given test is. For example, any code branch which is not executed during the test process is considered to not be covered by the test suite. Not only is this a good measure of test quality, it is a simple technique which allows the team to find uncovered code paths and fill in the gaps.

Where relevant, application invariants are listed along side the test cases for that section of code. Automated unit tests and user-driven integration tests validate that those invariants are maintained.

Acknowledgment

The team would like to thank Eric Credeur and Jamie Juneau from Techneaux Tech for their excellent mentorship, Dr. Kumar for his dedication and advice, and Ms. Devin Segue for her valuable feedback patience. The project would not have succeeded without each of them supporting us.

References

[1] Withers, Chris. Working with Excel Files in Python [Online]. Available: http://www.python-excel.org/

[2] Quotas - Google App Engine -- Google Cloud Platform [Online]. Available: https://cloud.google.com/appengine/docs/quotas