



GRADEBADGE  
DEVELOPMENT OF A CLOUD-BASED REWARD APPLICATION

---

A Project  
Presented to the  
Faculty of  
California State University,  
San Bernardino

---

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science  
in  
Computer Science

---

by  
Erwin Toni Soekianto  
April 2010

GRADEBADGE  
DEVELOPMENT OF A CLOUD-BASED REWARD APPLICATION

---

A Project  
Presented to the  
Faculty of  
California State University,  
San Bernardino

---

by  
Erwin Toni Soekianto

April 2010

Approved by:

\_\_\_\_\_  
David Turner, Chair, Computer Science and  
Engineering

\_\_\_\_\_  
Date

\_\_\_\_\_  
Richard J. Botting

\_\_\_\_\_  
Arturo I. Concepcion

© 2013 Erwin Toni Soekianto

## ABSTRACT

The purpose of this project is to investigate the use of cloud-based service to deliver cutting-edge application. For this purpose, the prototype of reward application using badges will be developed to illustrate the emerging paradigms, including the Node.js, NoSQL database using MongoDB, cloud computing using Heroku, and mobile devices.

In addition to Node.js, web technologies might be used such as HTML (5), Javascript, and CSS, and jQuery Mobile to serve as clients to this application. We would also be using cloud-based source control and repositories, GitHub, which would allow automation in deploying the application.

The goal of this application is to help any organizations or groups to interact with their members in the fun way. It would keep the members engaged by giving badges as rewards for effort or achievement they have done. In order to get this application going viral quickly; it would be integrated to social networking sites like Facebook.

## ACKNOWLEDGEMENTS

I would like to thank all the people with whom I have worked while pursuing my master's degree at California State University, San Bernardino (CSUSB). I wish I could list all their names but the list would be too long and I would still probably leave some people out. Studying in the School of Computer Science and Engineering at CSUSB has been a tremendous learning experience, both personally and professionally.

Thank you to the following faculty of the Computer Science and Engineering department for their invaluable guidance, advice, support, help, and patience during this project's long gestation: Dr. David Turner, Dr. Arturo Concepcion and Dr. Richard Botting.

## TABLE OF CONTENTS

<i>Abstract</i> . . . . .	iii
<i>Acknowledgements</i> . . . . .	iv
<i>List of Tables</i> . . . . .	viii
<i>List of Figures</i> . . . . .	ix
<i>1. Introduction</i> . . . . .	1
1.1 Background . . . . .	1
1.2 Facebook . . . . .	1
1.3 Heroku . . . . .	1
1.4 MongoDB . . . . .	2
1.5 MongoLab . . . . .	2
1.6 Git . . . . .	2
1.7 Bootstrap . . . . .	3
1.8 JQuery . . . . .	3
1.9 Node.js . . . . .	3
1.10 Purpose . . . . .	3
1.11 Project Scope . . . . .	3
1.12 Related Work . . . . .	4
1.13 Project Limitations . . . . .	4
1.14 Definitions, Acronyms, and Abbreviations . . . . .	4

2. <i>System Architecture</i> . . . . .	7
2.1 Overview . . . . .	7
2.2 Deployment Workflow . . . . .	7
2.3 Heroku . . . . .	8
2.4 MongoLab . . . . .	8
2.5 Facebook . . . . .	8
2.6 Client Browser . . . . .	8
3. <i>System Design</i> . . . . .	9
3.1 Design Overview . . . . .	9
3.2 Model View Controller Architecture . . . . .	9
3.3 Server-side Architecture Design . . . . .	9
3.4 Mapping of Model Classes to MongoDB . . . . .	11
3.5 Request Handler Operation . . . . .	12
3.6 Client-side Architecture Design . . . . .	12
3.7 System Interfaces . . . . .	12
3.8 Product Functions . . . . .	12
4. <i>Database Design</i> . . . . .	13
4.1 MongoDB . . . . .	13
4.2 MongoLab . . . . .	13
4.3 Documents . . . . .	13
4.4 Collections . . . . .	13
5. <i>Project Implementation</i> . . . . .	14
5.1 Welcome Screen . . . . .	15
5.2 Login Screen . . . . .	16
5.3 Quarter Screen . . . . .	17



5.4	Courses Screen . . . . .	18
5.5	Add Course Screen . . . . .	19
5.6	Course Details Screen . . . . .	20
5.7	Edit Course Screen . . . . .	21
5.8	Edit Course Name Screen . . . . .	22
5.9	Instructors Screen . . . . .	23
5.10	Add Insructor . . . . .	24
5.11	Delete Insructor . . . . .	25
5.12	Students . . . . .	26
5.13	Add Student . . . . .	27
5.14	Add Multiple Students . . . . .	28
5.15	Edit Student . . . . .	29
5.16	Grades . . . . .	30
5.17	Edit Grades . . . . .	31
5.18	Edit Student Grade . . . . .	32
5.19	Gradable Components . . . . .	33
5.20	Add Gradable Component . . . . .	34
5.21	Edit Gradable Component . . . . .	35
6.	<i>Conclusion and Future Direction</i> . . . . .	36
6.1	Conclusion . . . . .	36
6.2	Future Direction . . . . .	36
	<i>APPENDIX A: SERVER SOURCE CODE</i> . . . . .	37
	<i>APPENDIX B: CLIENT SOURCE CODE</i> . . . . .	39
	<i>References</i> . . . . .	41

## LIST OF TABLES

## LIST OF FIGURES

5.1	GradeBoard Welcome Screen . . . . .	15
5.2	Login Screen . . . . .	16
5.3	Quarter Screen . . . . .	17
5.4	Courses Screen . . . . .	18
5.5	Add Course Screen . . . . .	19
5.6	Course Details Screen . . . . .	20
5.7	Edit Course Screen . . . . .	21
5.8	Edit Course Name Screen . . . . .	22
5.9	Instructors Screen . . . . .	23
5.10	Add Instructor Screen . . . . .	24
5.11	Delete Instructor Screen . . . . .	25
5.12	Students Screen . . . . .	26
5.13	Add Student Screen . . . . .	27
5.14	Add Multiple Students Screen . . . . .	28
5.15	Edit Student Screen . . . . .	29
5.16	Grades Screen . . . . .	30
5.17	Edit Grades Screen . . . . .	31
5.18	Edit Student Grade Screen . . . . .	32
5.19	Gradable Component Screen . . . . .	33
5.20	Add Gradable Component Screen . . . . .	34
5.21	Edit Gradable Component Screen . . . . .	35

## 1. INTRODUCTION

### *1.1 Background*

A long time ago, businesses used to produce their own electric power. And due to engineering breakthrough in electric generator and transmission method, it became easier to produce and transmit electricity, to supply businesses that once produced their own electricity. As more businesses started buying electric power, making utility expanded and electricity cheaper.

And today, just like the utilities, instead of buying servers to run your websites or applications, you rent servers or server spaces from cloud computing providers. Just like renting an apartment, even you are in the same building with other people, you still have your own space. As more people rent and buy computing power, making cloud computing expanded and

Continue with the stories till cloud computing points.

The following describe the cloud computing service providers used in this project. Heroku, MongoLab, and GitHub.

### *1.2 Facebook*

### *1.3 Heroku*

Cloud computing is a model which makes use of computer hardware and software that are accessed through the Internet as services. There are several choices of cloud computing services available, but for this project we choose the one provided by

Heroku, the cloud computing partner of Facebook. [11].

The reasons are that Heroku lets you use and publish an application that people can use right away with no cost and obligation, and you can take advantage of the same scalable technologies that Facebook applications are built on, and attain a similar level of reliability, performance and security.

#### *1.4 MongoDB*

There are many different types of cloud-based datastore services to choose from. For this project we will use MongoDB, as it works well Node.js and Heroku. MongoDB is a scalable, high-performance, open source, NoSQL document-based database. MongoDB features include document-oriented storage, indexes, replication, high availability, auto-sharding, and querying.

#### *1.5 MongoLab*

#### *1.6 Git*

Git is a distributed version control system. This project uses git with GitHub, a cloud-based provider of remote git repository storage. Heroku uses git as a means to deploy web applications to its servers. Git allows easy creation of testing, staging, and production versions of the application.

## *1.7 Bootstrap*

## *1.8 JQuery*

## *1.9 Node.js*

Cloud-based services support apps written in several different programming languages, such as Java, Python, PHP, Javascript, Ruby and many more. For this project we would use Javascript running in a Node.js context. Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

## *1.10 Purpose*

To explore the new technologies, to create cross-platform reward application that individual can use.

## *1.11 Project Scope*

Project does not include database sharding features to allow greater degree of scalability.

The GradeBadge application provides the following functionalities:

- Create group
- Create Badge
- Add Member
- Issue Badges to Members

- View Badge Earned
- Share Badge to Social Networking

### *1.12 Related Work*

Explain manoj work and explain the differences, Google App Engine VS Heroku, Google Data Store VS MongoLab, Java VS Node.js, JQuery Mobile VS Bootstrap. [33]

### *1.13 Project Limitations*

Users must have Facebook account, logged in to facebook and authorized access to basic information (name, profile picture and friend list) .For best experience must use modern browser in either PC or tablet or smart phone.

### *1.14 Definitions, Acronyms, and Abbreviations*

The definitions, acronyms, and abbreviations used in the document are described in this section.

- GradeBadge: The name of this project
- API: Application Programming Interface is a set of routines that an application uses to request and carry out low-level services performed by a computer's operating system; also, a set of calling conventions in programming that defines how a service is invoked through the application [9].
- Cloud computing: Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet) [6].

- JQuery: A javascript library provided by JQuery for building web based applications [19].
- UI: User Interface
- CSUSB: California State University, San Bernardino.
- HTML: HyperText Markup Language is the authoring language used to create documents on the World Wide Web [29].
- HTTPS: Hyper Text Transfer Protocol Secure is a secure network protocol used to encrypt data transferred between server and client [13].
- MVC: Model-View-Controller is an architectural pattern used in software engineering to isolate business logic from user interface considerations [21].
- UML: The Unified Modeling Language is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems [28].
- Microsoft Azure: Cloud Computing platform provided by Microsoft [32].
- Amazon Web Services: Cloud Computing platform provided by Amazon [3].
- Heroku: Cloud Application platform provided by Heroku [11].
- Android : Mobile Operating System provided by Google [4].
- IOS : Mobile Operating System provided by Apple [15].
- PhoneGap: Open Source Framework for creating Mobile Apps [26].
- NoSQL: Uses key-value pairs for storing data unlike traditional Relational Database Management [22].
- JSON : Javascript Object Notation built using key and value pairs [20].



- Ajax: Asynchronous JavaScript and XML/JSON format for communicating from client to the server [2].
- OOP : Object Oriented Programming concept with objects representing real world entities. Methods expose state of the object [31].

## 2. SYSTEM ARCHITECTURE

### 2.1 *Overview*

This application uses HTTPS exclusively for security reason, except in local developer environment we use HTTP.

### 2.2 *Deployment Workflow*

There are three type of environments used in the deployment workflow: Development, Staging and Production. And this is how the workflow will look like:

- Developers work on new features or bugs fixing in development branch. Only minor updates are committed directly to stable development branch.
- Once features are implemented and/or set of bugs are fixed, they are merged in to staging branch and deployed to staging environment for testing and quality assurance
- After testing is completed, the snapshot of staging branch is kept for prduction deployment, otherwise the process will repeat until the testing is completed.
- On the release date, the working staging branch is deployed to production environment.

On this project, git is used as code repositories, to manage developments, staging and production branch. And Heroku toolbelt is also used to set the enviroment config

variable for each deployment. Heroku allows users to use git to deploy automatically from local repositories.

### *2.3 Heroku*

In this project there are two sets of heroku instance used, staging and production. Heroku connects to MongoLab using Mongo Protocol to get and/or write the data to database, and Heroku also talks to Facebook server via Open Graph API.

### *2.4 MongoLab*

In this project there are three sets of mongo database used, development, staging and production. It is important to keep the versions of database since new version of changes may include changes in database structure, so rolling back or forward the application version would not cause any error.

### *2.5 Facebook*

Facebook is playing an important role in this project. Facebook provides user authentication and social media integration. Facebook allows connection using Facebook API and Open Graph API.

### *2.6 Client Browser*

Client browser uses HTTPS GET for static content, and HTTPS POST for AJAX request to Heroku. And client browser also connects to Facebook server directly using Facebook API and Open Graph API in HTTPS.

### 3. SYSTEM DESIGN

#### 3.1 *Design Overview*

All incoming AJAX requests are submitted using HTTP POST and contain data encoded using JSON.....

#### 3.2 *Model View Controller Architecture*

This application is based on Model View Controller Architecture..... There are two sections of this project, Server-side and Client-side...

#### 3.3 *Server-side Architecture Design*

All node js modules that start with req-\*.js are request handler that get requested from router.js. All ajax requests will go through req\_op.js, that verifies the user logged-in to Facebook and app\_version is current. Every req\_op.js request must contains Facebook access\_token and app\_version. If the user is not logged-in to Facebook, req\_op.js returns the following JSON document, login:true, if the version is not current, then req\_op.js return the following JSON document, ver:true.

- .env : This is the setup file that contains environment variables. This file only exist in developer local environment, and these values in this file would be set in each Heroku environment config for staging and production.
- .gitignore: This is the setup file that contains list of files or folders that will be

ignored when committing or pushing to git repositories.

- `.slugignore`: This is the setup file that contains list of files or folders that will be ignored when calculating the slug limit in Heroku
- `package.json`: This is the setup file that contains of list of dependencies and engine version use in the application. This file also contains application name, version and description.
- `Procfile`: This is the setup file that tells Heroku how to launch the application
- `main.js`: This is the main module in node js, which contains the code that verify all neccessary environment variables are set correctly. It also invoke initialization in neccesarry modules to start the application, after the initialization is completed, it starts the HTTP request handling loop.
- `router.js`: This module routes incoming requests to the right module.
- `app_ajax.js`: This module contains the application wide AJAX handling routines
- `app_http.js`: This module contains all of HTTP protocol routines for the application, caching headers, compression header and other HTTP based optimization are implemented in this module.
- `fb.js`: This module contains all code that interact with Facebook.
- `logger.js`: This module contains application wide logging functionalities.
- `model.js`: This module initializes the database connnection pool during server start
- `req_app.js`: This module handles request for application HTML template for badge earner
- `req_counter.js`: This module handles request for logging counter

- req\_file.js: This module handles request for static content.
- req\_issuer.js: This module handles request for application HTML template for badge issuers
- req\_mem.js: This module handles request for memory usage
- req\_root.js: This module handles request for static content under the root URL
- req\_op.js : This module handles all AJAX request from client and routes to appropriate modules.

### *3.4 Mapping of Model Classes to MongoDB*

There will be one node js module to represent the mongoDB collection, named model\_(collection\_name).js. And many-to-many relationships are represented by linking documents, named (a)\_(b)\_links

- model\_group.js: this node.js module represents Groups collection
- model\_badge.js: this node.js module represents Badges collection
- model\_user.js: this node.js module represents Users collection
- model\_group\_admin.js: this node.js module represents group\_admin\_links collection
- model\_group\_member.js: this node.js module represents group\_member\_links collection
- model\_user\_badge.js: this node.js module represents user\_badge\_links collection
- model\_group\_badge.js: this node.js module represents group\_badge\_links collection

### 3.5 Request Handler Operation

There will be one node js module to handle ajax request from client, named `op_(request).js`.

Every request may read, write or update to and from more than one collection

- `op_read_badges_by_group.js`
- `op_read_groups_by_admin.js`
- `op_save_badge.js`
- `op_save_group.js`
- still more.....

### 3.6 Client-side Architecutre Design

- `app.html` : This is the html template for badge earner page
- `issuer.html` : This is the html template for badge issuer page
- `public_root/channel.html` : This is the static content required by Facebook
- `public_root/favicon.ico` : This is the statuc content for icon use in the browser
- `public_ver/app.js` : This is the client java-script
- `public_ver/style.css`: This is the css file use
- still more.....

### 3.7 System Interfaces

### 3.8 Product Functions

## 4. DATABASE DESIGN

Explain Document-oriented design

4.1 *MongoDB*

4.2 *MongoLab*

4.3 *Documents*

4.4 *Collections*

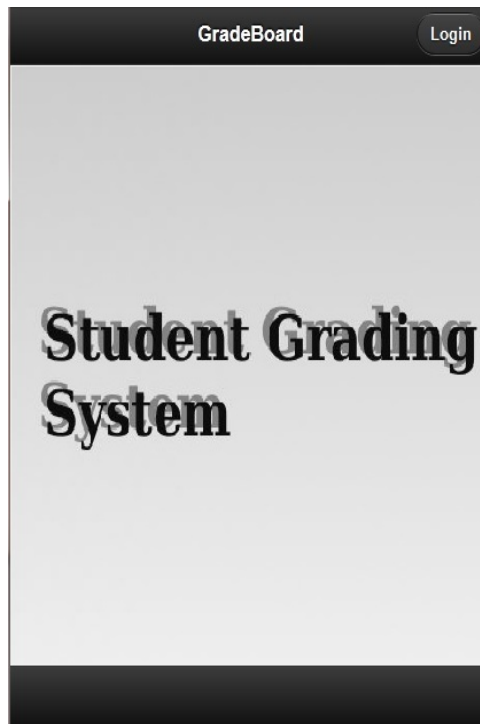


## 5. PROJECT IMPLEMENTATION

The GradeBadge application is designed to work on mobile devices and desktop computers. The UI of the application is developed using Bootstrap. When a page requires the data to be loaded from server or modified or deleted, a request is sent to the Web server over HTTP or HTTPS. The requests are sent to the Web server using Ajax. For handling Ajax requests and responses, this application uses JQuery Ajax API. All UI components are dynamically created or initialized in response to the data received from the Web server.

## 5.1 Welcome Screen

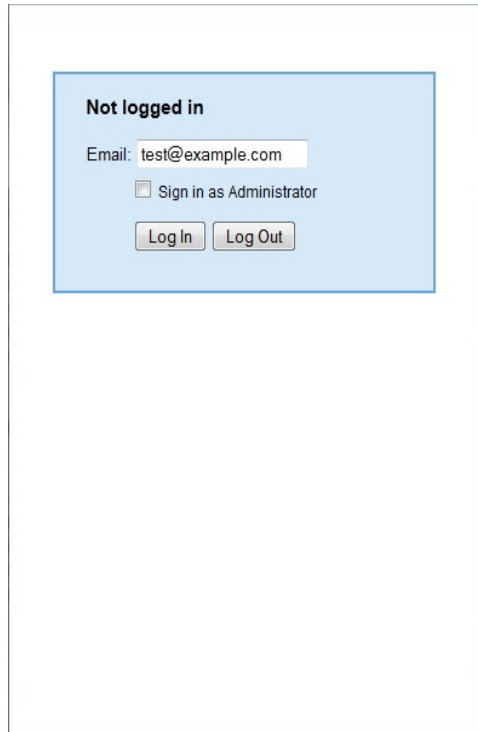
When the GradeBoard application is loaded, a welcome screen with login button is presented to the user as shown in the Figure 5.1. The welcome screen shows application logo and login button for users to login to the system.



*Fig. 5.1:* GradeBoard Welcome Screen

## 5.2 Login Screen

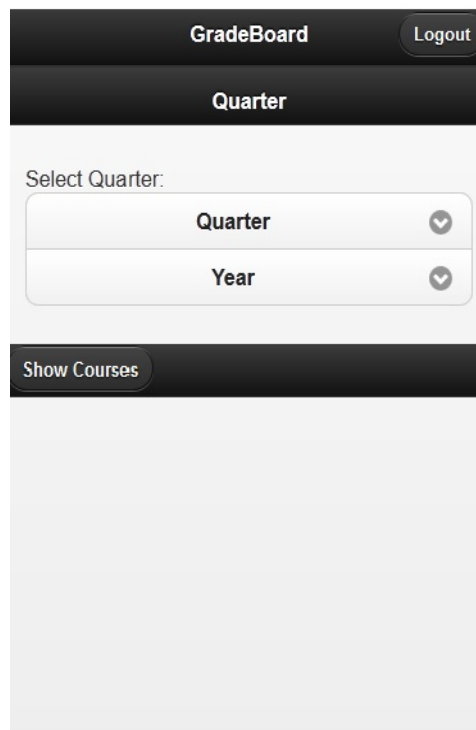
GradBoard uses Google accounts for authentication and authorization. When the user clicks on the login button, the screen is automatically redirected to the Google login screen as shown in Figure 5.2. There are two types of access users can get, namely admin and user. For admin access, the user is first authenticated by Google and then the user entry is checked in the admin table. For user access, the user account is authenticated by Google. After logging in, the quarter screen is shown.



*Fig. 5.2:* Login Screen

### 5.3 Quarter Screen

The quarter screen displays quarter and year information as shown in Figure 5.3. The quarter screen is shown after user logs into the system. After selecting quarter and year, the instructor clicks on the show courses button to view the list of courses.



The screenshot shows a mobile application interface for 'GradeBoard'. At the top, there is a dark header bar with the text 'GradeBoard' on the left and a 'Logout' button on the right. Below this is a section titled 'Quarter' in a dark bar. Underneath, the text 'Select Quarter:' is followed by two dropdown menus. The first dropdown is labeled 'Quarter' and the second is labeled 'Year', both with downward-pointing arrow icons. Below these dropdowns is a dark bar with a 'Show Courses' button. The bottom half of the screen is a large, empty light gray area.

Fig. 5.3: Quarter Screen

## 5.4 Courses Screen

The courses screen shows a list of courses as shown in Figure 5.4. Using the courses screen, the instructor can add a new course or view the details of the existing course. After the user clicks on the course, the course details page is loaded.

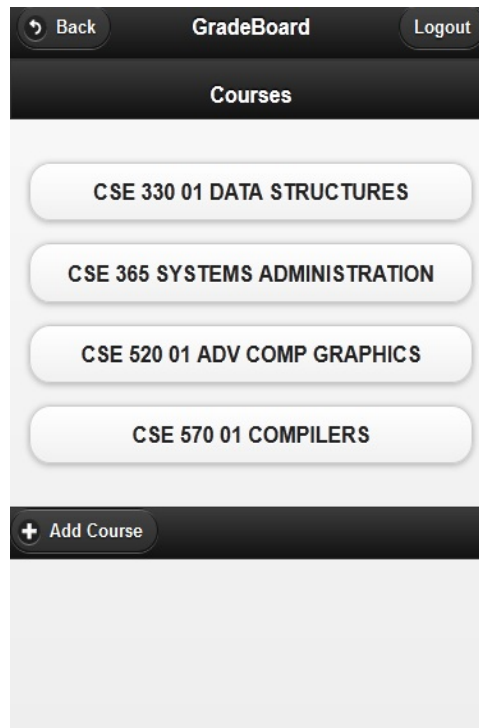
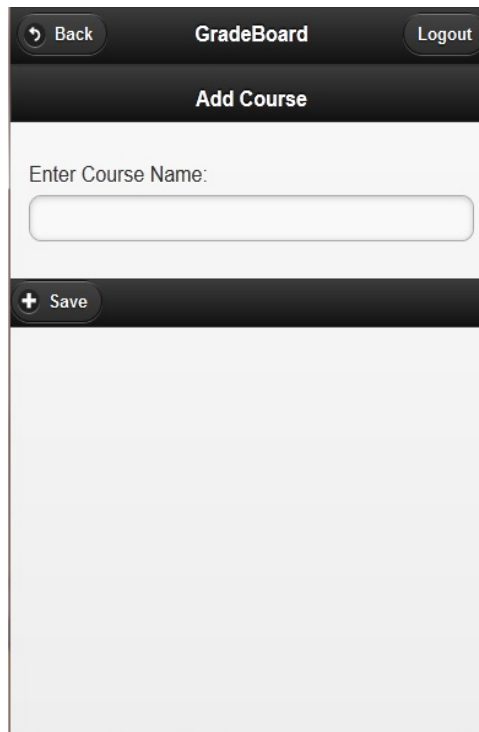


Fig. 5.4: Courses Screen

## 5.5 Add Course Screen

The add course screen is shown when the user clicks on the add button in the courses screen. Using this screen, instructors can add a new course as shown in Figure 5.5. This page checks that the course value entered already exists and displays an appropriate error message if not. The back button allows users to navigate to the course list page.



The image shows a mobile application screen titled "Add Course". At the top, there is a dark header bar with three buttons: "Back" (with a left arrow icon), "GradeBoard", and "Logout". Below the header, the title "Add Course" is centered. The main content area has a light gray background. It contains a text label "Enter Course Name:" followed by a white text input field with rounded corners. Below the input field, there is a dark button with a white plus icon and the text "Save". The bottom half of the screen is a large, empty light gray rectangular area.

Fig. 5.5: Add Course Screen

## 5.6 Course Details Screen

The course details screen allows instructors to view course details such as the number of students registered and the number of seats available. The course details screen is shown in Figure 5.6. From the course details screen, the user can edit the course, edit the grade components, or delete the course.

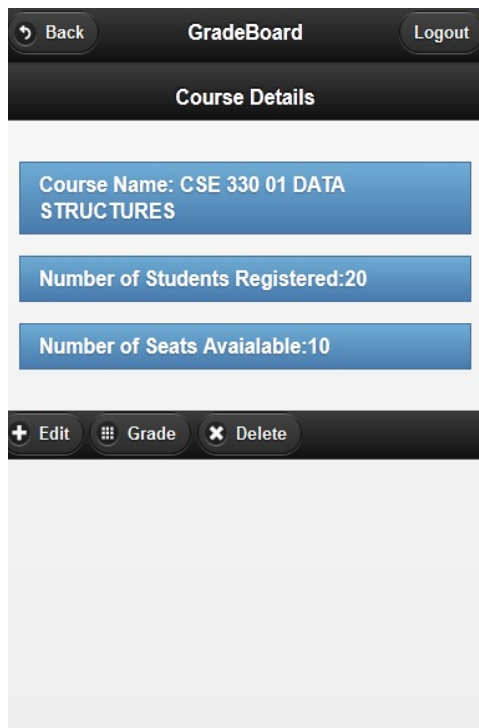


Fig. 5.6: Course Details Screen

## 5.7 Edit Course Screen

The edit course screen allows instructors to edit the course details as shown in Figure 5.7. The instructor can modify the course name, add additional instructors, add students and add gradable components.

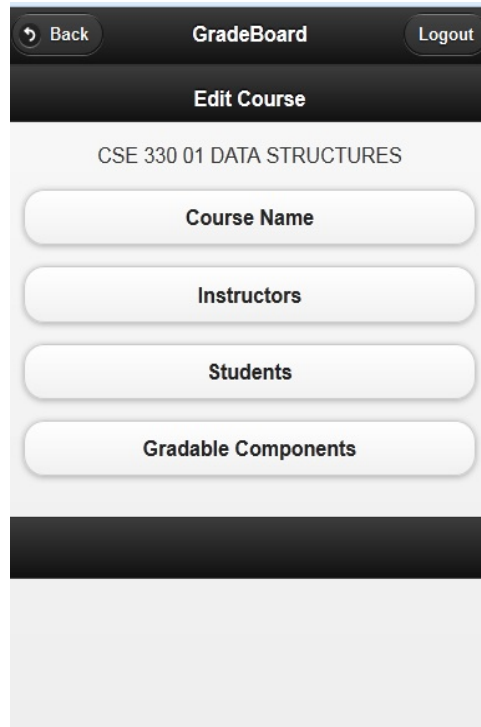


Fig. 5.7: Edit Course Screen



## 5.8 Edit Course Name Screen

The edit course name screen allows instructors to edit the course name as shown in Figure 5.8. A validation is added to check if the course name entered by the user is empty or if it already exists. Changing the course name involves passing the current course name and new course name to the server. The server searches first the course entity with name equal to the current course name and then replaces the course name property value with the new course name.

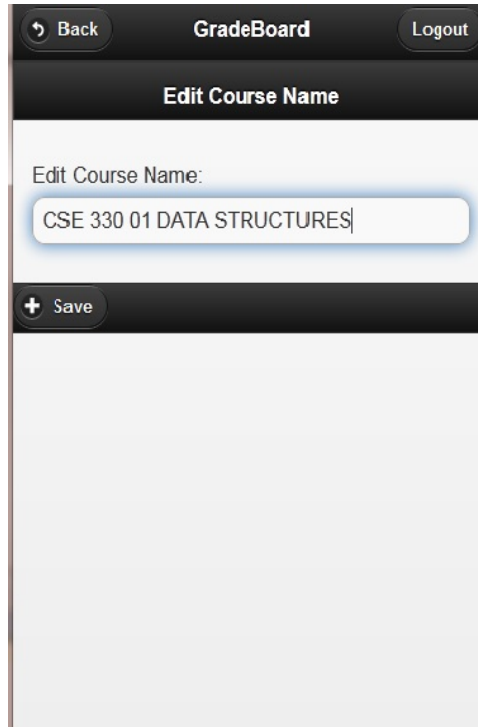


Fig. 5.8: Edit Course Name Screen

## 5.9 Instructors Screen

The instructors screen displays a list of instructors for a course as shown in Figure 5.9. A course can have multiple instructors. Internally, instructors are stored as auth objects with a parent instructor (who created the course). Each auth object will contain the id of the course entity.

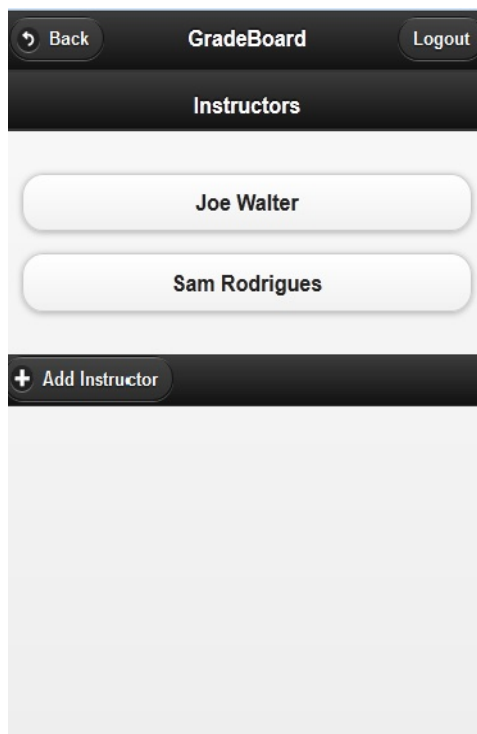
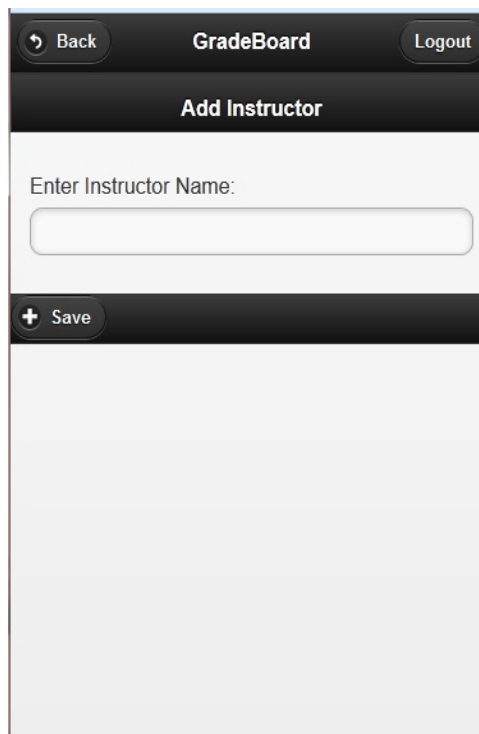


Fig. 5.9: Instructors Screen

## 5.10 Add Instructor

The add instructor screen allows an instructor to add another instructor as shown in Figure 5.10. Only the instructor who originally created the course can add additional instructors. The new instructors have restricted access. Permission to change the course name or gradable components is not given to new instructors.



The image shows a mobile application screen titled "Add Instructor". At the top, there is a dark header bar with three buttons: "Back" (with a left arrow icon), "GradeBoard", and "Logout". Below the header, the title "Add Instructor" is centered. The main content area has a light gray background. It contains the text "Enter Instructor Name:" followed by a white text input field with rounded corners. At the bottom of the form, there is a dark bar with a "+ Save" button.

Fig. 5.10: Add Instructor Screen

### 5.11 Delete Instructor

The delete instructor screen allows the original instructor to remove an instructor from the course as shown in Figure 5.11. The instructor who originally created the course can not be deleted, so there should always be at least one instructor for the course.

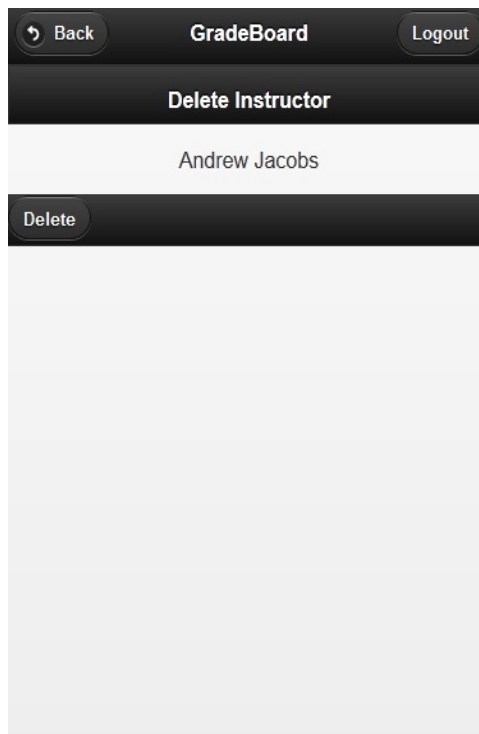


Fig. 5.11: Delete Instructor Screen

## 5.12 Students

The student screen lists all the students registered for the course as shown in Figure 5.12. New students can be added by clicking on the add student button. There is also an option to add multiple students at once.

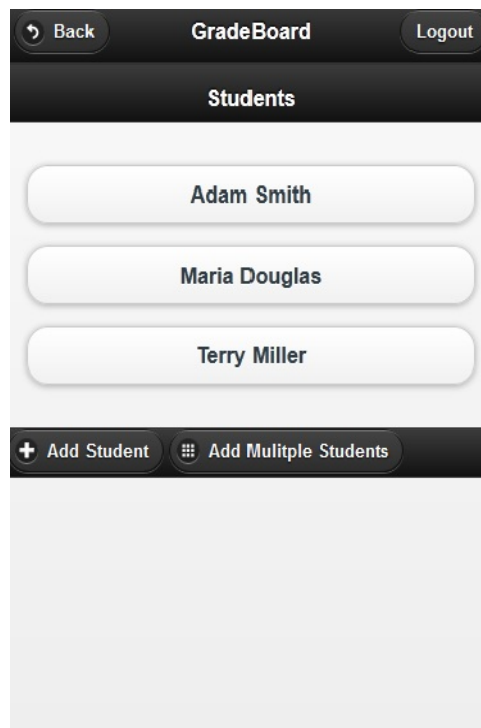
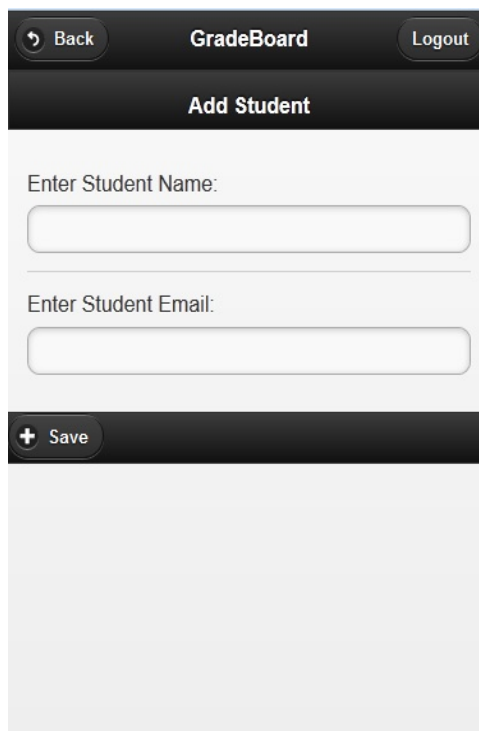


Fig. 5.12: Students Screen

### 5.13 Add Student

This screen allows instructors to add students to the course as shown in Figure 5.13. A validation is added to check for an empty student name or email address. Also, if a student with same name and email address already exists, an error message is shown to the instructor.

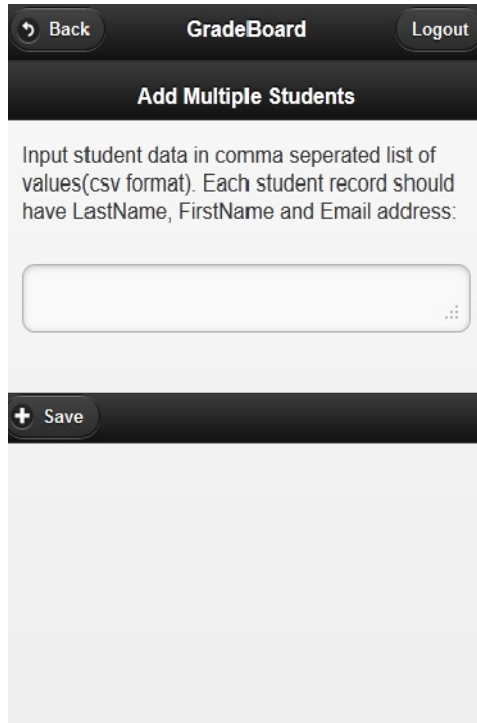


The image shows a mobile application screen titled "Add Student". At the top, there is a dark header bar with three buttons: "Back" (with a left arrow icon), "GradeBoard" (the app title), and "Logout". Below the header, the title "Add Student" is centered. The main content area has a light gray background and contains two text input fields. The first field is labeled "Enter Student Name:" and the second is labeled "Enter Student Email:". Both fields are empty. At the bottom of the form, there is a dark bar with a "+ Save" button. The bottom half of the screen is a large, empty light gray area.

Fig. 5.13: Add Student Screen

### 5.14 Add Multiple Students

The add multiple students screen allows instructors to enter multiple student data at once as shown in Figure 5.14. This helps the instructor to update student data at once when a course is created. To add multiple student data, instructors need to provide a comma separated list of student data in an HTML textarea field. The student data must contain three fields: first name, last name and email address. The instructor is notified if any student record is missing one of these fields.

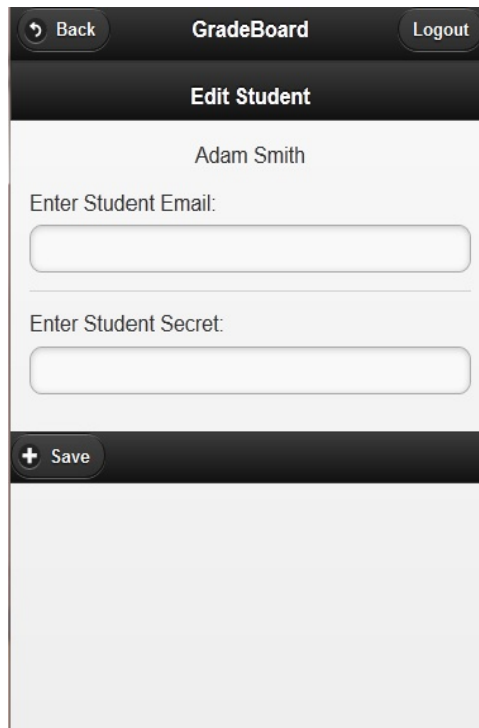


The screenshot shows a mobile application interface for 'GradeBoard'. At the top, there is a dark header bar with a 'Back' button (left arrow icon), the title 'GradeBoard' in the center, and a 'Logout' button (right arrow icon). Below the header is a section titled 'Add Multiple Students'. This section contains a text instruction: 'Input student data in comma seperated list of values(csv format). Each student record should have LastName, FirstName and Email address:'. Below the instruction is a large, empty text input field with rounded corners and a small '...' icon on the right side. At the bottom of the screen, there is a dark bar with a '+ Save' button.

Fig. 5.14: Add Multiple Students Screen

### 5.15 *Edit Student*

The edit student screen allows instructors to edit the details of a student as shown in Figure 5.15. All of the student information can be changed by the instructor.



The screenshot shows a mobile application interface for editing a student. At the top, there is a dark header bar with three buttons: a back arrow labeled "Back", the app name "GradeBoard" in the center, and a "Logout" button on the right. Below this header is a section titled "Edit Student" in white text on a dark background. The main content area has a light gray background and displays the name "Adam Smith" at the top. Below the name are two input fields: the first is labeled "Enter Student Email:" and the second is labeled "Enter Student Secret:". At the bottom of the form is a dark bar with a "+ Save" button. The bottom portion of the screen is a large, empty light gray area.

*Fig. 5.15:* Edit Student Screen



## 5.16 Grades

The grades screen shows an instructors list of the names of all gradable components as shown in Figure 5.16. If a new grade is required, then the instructor needs to add the gradable component in the add gradable component screen. Clicking on the grade button lists all the students registered for the course.

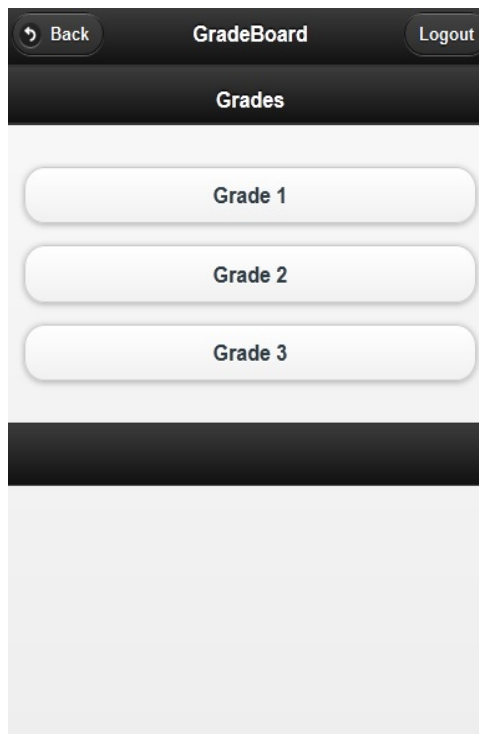


Fig. 5.16: Grades Screen

### 5.17 Edit Grades

The edit grade screen presents to the instructor a list of students whose grades can be edited as shown in Figure 5.17. Instructors can edit the individual grade by clicking on the student button. The view grade sheet option allows the instructor to view the grade sheet of the entire course.

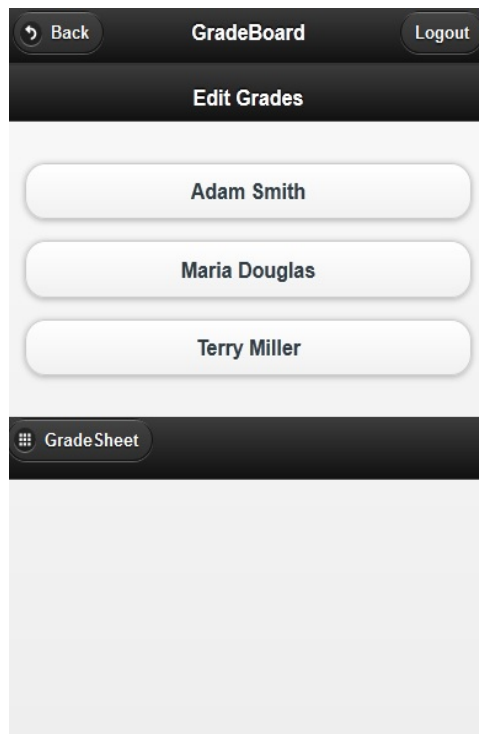
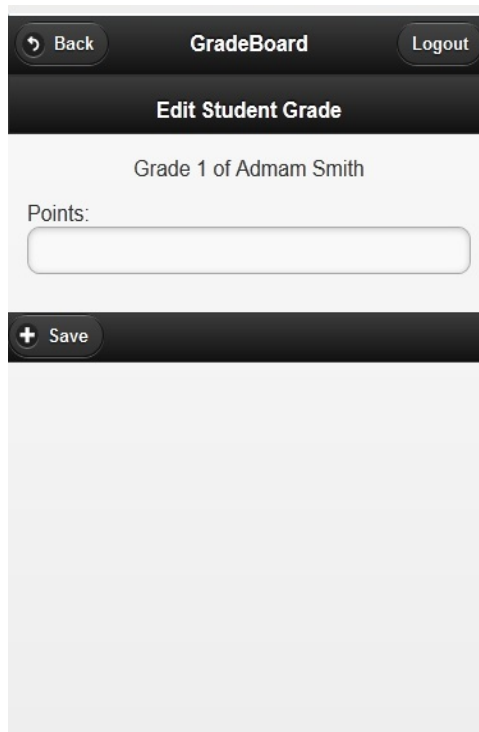


Fig. 5.17: Edit Grades Screen

### 5.18 *Edit Student Grade*

The edit student grade screen allows instructors to enter grade points of a student as shown in Figure 5.18. A validation is provided to check that values are not empty and do not exceed the maximum value of grade points entered in the gradable component.



The screenshot shows a mobile application interface for editing a student's grade. At the top, there is a dark header bar with three buttons: a 'Back' button with a left-pointing arrow, the 'GradeBoard' app name in the center, and a 'Logout' button. Below this header is a dark bar with the title 'Edit Student Grade'. The main content area is light gray and contains the text 'Grade 1 of Admam Smith'. Below this, the label 'Points:' is followed by a white rectangular input field with rounded corners. At the bottom of the screen, there is a dark bar with a single button labeled '+ Save'.

*Fig. 5.18:* Edit Student Grade Screen

### 5.19 Gradable Components

The gradable component screen displays a list of gradable components of a course as shown in Figure 5.19. A new gradable component can be added by clicking on the add gradable component button.

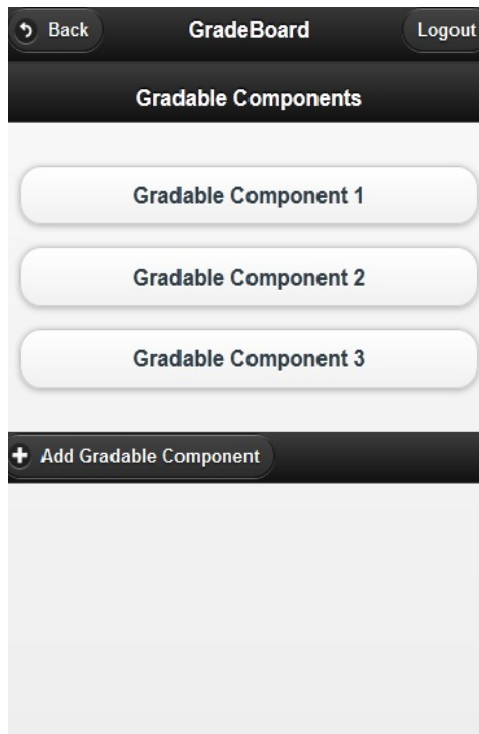
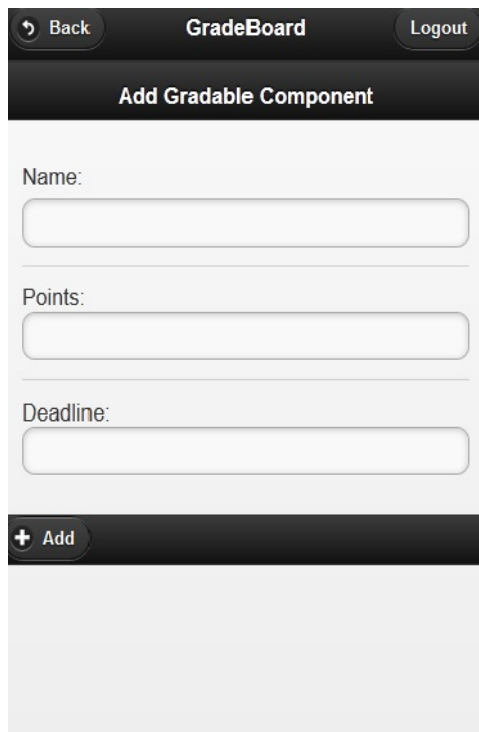


Fig. 5.19: Gradable Component Screen

## 5.20 Add Gradable Component

The add gradable component screen allows instructors to add a gradable component for the course as shown in Figure 5.20. Name, points and deadline are required fields and a validation is added to check for empty values. A check for duplicated entries is added to make sure gradable components are unique in a course.

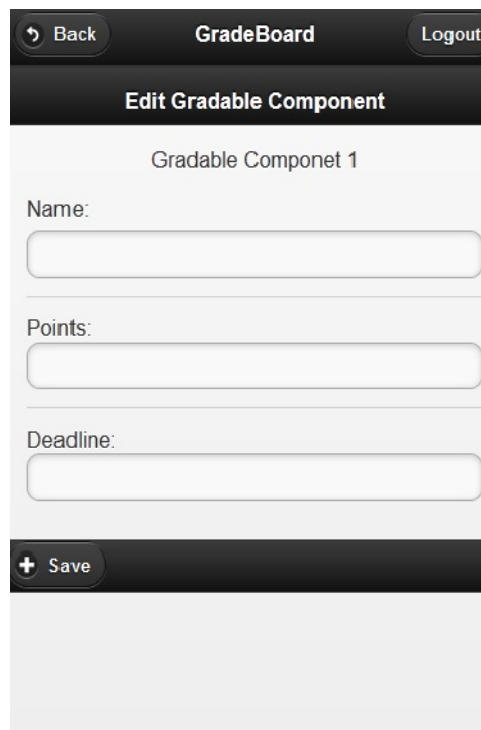


The screenshot shows a mobile application interface for 'GradeBoard'. At the top, there is a dark header bar with a 'Back' button on the left, the 'GradeBoard' title in the center, and a 'Logout' button on the right. Below this header is a dark bar with the title 'Add Gradable Component'. The main content area is light gray and contains three input fields, each with a label above it: 'Name:', 'Points:', and 'Deadline:'. Each label is followed by a white rounded rectangular input field. At the bottom of the form, there is a dark bar with a white plus icon and the text '+ Add'. Below this bar is a large, empty light gray rectangular area.

Fig. 5.20: Add Gradable Component Screen

## 5.21 Edit Gradable Component

The edit gradable component screen allows instructors to edit gradable components for the course as shown in Figure 5.21. Name, points and deadline are required fields, so a validation is performed to check for empty values.



The screenshot shows a mobile application interface for editing a gradable component. At the top, there is a dark header bar with a 'Back' button on the left, the title 'GradeBoard' in the center, and a 'Logout' button on the right. Below this header is a dark bar with the title 'Edit Gradable Component'. The main content area is light gray and contains the text 'Gradable Component 1' at the top. Below this, there are three input fields, each preceded by a label: 'Name:', 'Points:', and 'Deadline:'. Each input field is a rounded rectangle with a thin border. At the bottom of the form, there is a dark bar with a '+ Save' button. The bottom of the screen is a light gray area.

Fig. 5.21: Edit Gradable Component Screen

## 6. CONCLUSION AND FUTURE DIRECTION

### 6.1 *Conclusion*

Conclusion here

### 6.2 *Future Direction*

Future Direction here

- Provide a responsive design using technologies such as Bootstrap to support dynamic screen sizes on multiple devices and desktop systems [5].
- item here
- Add new interfaces to the data model to support alternative database systems that could be used at lower cost, such as MongoDB [1].

APPENDIX A

SERVER SOURCE CODE



```
//Auth.java
```

```
//Course.java
```

```
//Instructor.java:
```

```
\\InstructorControllerServlet.java:
```

```
\\LoginFilter.java:
```

```
// LogoutFilter.java:
```

APPENDIX B

CLIENT SOURCE CODE

```
// app.js
```

## REFERENCES

- [1] Agile and scalable. <http://www.mongodb.org/>.
- [2] Ajax (programming). [http://en.wikipedia.org/wiki/Ajax\\_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming)).
- [3] Amazon Web Services. <http://aws.amazon.com/>.
- [4] Android Mobile Computing Platform.  
<http://developer.android.com/about/index.html>.
- [5] Bootstrap. <http://twitter.github.com/bootstrap/>.
- [6] Cloud Computing Document.  
[http://www.en.wikipedia.org/wiki/Cloud\\\_computing](http://www.en.wikipedia.org/wiki/Cloud\_computing).
- [7] Eclipse Founaction. <http://www.eclipse.org/>.
- [8] Entities, Properties, and Keys.  
<https://developers.google.com/appengine/docs/java/datastore/entities>.
- [9] Glossary. <http://technet.microsoft.com/en-us/library/bb742416.aspx>.
- [10] Google Developers Academy.  
<https://developers.google.com/appengine/training/intro/whatisgae/>.
- [11] Heroku How It Works. <http://www.heroku.com/how>.
- [12] How Entities and Indexes are Stored.  
[https://developers.google.com/appengine/articles/storage\\_breakdown](https://developers.google.com/appengine/articles/storage_breakdown).
- [13] Http secure. <http://en.wikipedia.org/wiki/HTTPS>.
- [14] Index Definition and Structure.  
<https://developers.google.com/appengine/docs/python/datastore/indexes>.

- [15] IOS Mobile Computing Platform. <http://en.wikipedia.org/wiki/IOS>.
- [16] Java Database Connectivity.  
<http://docs.oracle.com/javase/6/docs/technotes/guides/jdbc>.
- [17] Jetty (web server). [http://en.wikipedia.org/wiki/Jetty\\_\(web\\_server\)](http://en.wikipedia.org/wiki/Jetty_(web_server)).
- [18] JQuery Mobile 1.2 Reference Document.  
<http://www.jquerymobile.com/demos/1.2.0/docs/about/features.html>.
- [19] JQuery Official Document. <http://www.jquery.com/>.
- [20] Json. <http://en.wikipedia.org/wiki/Json>.
- [21] Model view controller. <http://en.wikipedia.org/wiki/Model-view-controller>.
- [22] Nosql. <http://en.wikipedia.org/wiki/NoSQL/>.
- [23] Official Documentation of Google App Engine Java Datastore.  
<https://developers.google.com/appengine/docs/java/datastore/>.
- [24] Official Documentation of Java Servlet Technology.  
<http://www.oracle.com/technetwork/java/whitepaper-135196.html>.
- [25] Official Java Technology Document.  
[http://www.java.com/en/download/faq/whatis\\_java.xml](http://www.java.com/en/download/faq/whatis_java.xml).
- [26] PhoneGap. <http://phonegap.com/>.
- [27] PolyModel.  
<https://developers.google.com/appengine/docs/python/ndb/polymodelclass>.
- [28] Unified Modeling Language. <http://www-01.ibm.com/software/rational/uml/>.
- [29] W3C community. <http://www.w3.org/TR/REC-html40/>.
- [30] Webopedia Definitions And Terms.  
<http://www.webopedia.com/TERM/J/J2EE.html/>.
- [31] What is an Object?  
<http://docs.oracle.com/javase/tutorial/java/concepts/object.html>.

- [32] Windows Azure. <http://www.windowsazure.com/en-us/develop/overview/>.
- [33] Manoj Kulkarni. *GradeBoard: A Cloud-Based Solution for a Student Grading System*. CSUSB - Masters Project, 2013.