iAttend:

Design Document 2, Draft i

Three Furious Locomotives

March 25, 2020

iAttend Design Document 2

DD2-ThreeFuriousLocomotives.tex
Draft i
March 25, 2020
Three Furious Locomotives

Revisions

Version	Primary Author	Description of Version	Date Completed
i	Matthew Dutt, Alec Barker, and Michael Jarman	Initial creation of document.	3/25/20

Contents

1 Introduction **4**
- 1.1 Overview . . . 4
- 1.2 List of Deliverables, with delivery dates . . . 4
- 1.3 Dates of scheduled reviews of plan and project schedule . . . 4
- 1.4 Glossary of Terms . . . 4

2 Module 1 - Create an Account **6**
- 2.1 Classification . . . 6
- 2.2 Purpose . . . 6
- 2.3 Constraints . . . 6
- 2.4 Interactions . . . 6
- 2.5 Processing . . . 6
- 2.6 Data . . . 6
- 2.7 Relevant Diagrams for Module 1 . . . 7

3 Module 2 - Password Reset **8**
- 3.1 Classification . . . 8
- 3.2 Purpose . . . 8
- 3.3 Constraints . . . 8
- 3.4 Interactions . . . 8
- 3.5 Processing . . . 8
- 3.6 Data . . . 8
- 3.7 Relevant Diagrams for Module 2 . . . 9

4 Module 3 - Database Tables **10**
- 4.1 Classification . . . 10
- 4.2 Purpose . . . 10
- 4.3 Constraints . . . 10
- 4.4 Interactions . . . 10
- 4.5 Processing . . . 10
- 4.6 Data . . . 10
- 4.7 Relevant Diagrams for Module 3 . . . 11

5 Module 4 - Get Attendance **12**
- 5.1 Classification . . . 12
- 5.2 Purpose . . . 12
- 5.3 Constraints . . . 12
- 5.4 Interactions . . . 12
- 5.5 Processing . . . 12

5.6 Data 12
5.7 Relevant Diagrams for Module 4 13

1 Introduction

1.1 Overview

The development team affiliated with Mount Union's 2020 CSC 492 Software Engineering course known as the Three Furious Locomotives is building a system for the Music department that will replace the current manual method of recording attendance with a software solution that will simplify and speed up the process of recording attendance. The product, which will be known as iAttend, will also allow the designation of requirements, which will allow progress to be tracked, and reports to be easily generated. Using the system will be as easy. After the easy account creation and setup process, all the admin and professor roles need to do is scan QR codes to record attendance. This will be akin to scanning goods at a self-checkout at a grocery store. Those who are less confident will find iAttend easy to learn and use, since its learning curve is low. Those concerned about security need not worry, since security has been a top priority of the development team since the project's conception.

The purpose of this document is to outline the design of the iAttend software. This document will provide high level design detail on a small selection of modules that are paramount to the functionality of the software product.

1.2 List of Deliverables, with delivery dates

Name	Date
Project Selection	01/15/2020
Create Blog	02/05/2020
Requirements Specification Document and Press Release 1	02/10/2020
Work Schedule	02/17/2020
Presentation 1	02/19/2020
Design Document 1	03/18/2020
Design Document 2	04/01/2020
Presentation 2/Prototype	04/06/2020
Press Release 2	04/15/2020
User Manual	04/27/2020
Project Due	04/29/2020

1.3 Dates of scheduled reviews of plan and project schedule

The development team will meet at various points before the project deadline to review the progress of the project and the plan to develop it. The group will meet bi-weekly to review the project starting on March 16. As the due date draws closer, it will move to weekly meetings. This means that the team will review their schedule and project design on March 16 and 30, and April 13, 20, and 27.

1.4 Glossary of Terms

- "The Software" shall refer the iAttend system in its entirety, including all devices involved.
- "The Team" shall refer the software development team in charge of creating The Software, known as Three Furious Locomotives.
- "The Administrator (Admin)" shall refer the owner of an iAttend Administrator account, which will have permission change configuration of their course as set up in the iAttend portal. For example, administrators would be department chairs or a professor configuring iAttend for their own classroom use.
- "The Moderator (Mod)" shall refer a user with elevated permissions. This user does not have permission change configuration of an established course but is able scan users and manually change attendance records. Professors assisting The Administrator with scanning would be examples of moderators.

- "The User (User)" shall refer a user with an ID that has no permissions beyond viewing their own attendance records. E.g. students, attendees.
- "The Portal" shall refer the web site that is accessible via a web browser. It will allow all types of users log in and view/edit records depending on their permission levels.
- "Course" shall refer the environment set up by The Administrator. It includes a roster of users record attendance for and has its own set of attendance records for each user for each event that takes place in that course. This could be a Math 101 course in Spring of 2020, for instance.
- "Check-In" shall refer the act of a user being scanned by a moderator or administrator, thereby recording attendance. This process involves the scanner scanning the QR code, identifying The User, and recording that attendance affirmation the database for the course.
- "Event" shall refer the specific event within the course for which a set of check-ins are grouped by. This could be attendance for the third lecture of a course, for instance.
- "The App" shall refer the mobile application for iOS primarily responsible for scanning QR codes and sending the check-in the database.
- Personally Identifiable Information (PII) refers information about a user that makes her/him identifiable in the real world, such as name, address, birthday, etc.
- Secure Sockets Layer (SSL) refers a way that data is securely sent across the Internet using encryption.
- Public Key Encryption refers a way of encrypting data where one key is publicly available while another key is privately held by the software owner. Together, these keys allow for encryption and decryption of data.
- Simple Mail Transfer Protocol (SMTP) is a way of transmitting emails.
- A Plaintext value is a value that can be read and understood by humans. It is the opposite of an encrypted value.
- A Hashed value is the encrypted version of a value such as a password.
- A Salt value is a string of characters that is used by an encryption algorithm to hash another value.

2 Module 1 - Create an Account

2.1 Classification

Creating an account is a module consisting of multiple functions that will verify that the account is available, store the new account in the database, and verify that the user has access to their email.

2.2 Purpose

This module consists of the process for a user to create an account. The user will enter account information into the application and verify the account setup by answering an email, and their account will be stored in the database and become accessible to the user.

2.3 Constraints

This process should be quick for the user. There should be minimal delay between signing up and receiving the verification email, as well as minimal delay between verifying the email and being able to sign into an account. The information must be stored in the database, so there should be a process for ensuring that the user does not enter malicious information such as SQL injections or invalid information such as using the same email as an existing account. This module should only be triggered by user events. The information that the user inputs should not be stored within the web program itself. Instead, the web program should encrypt the necessary data and send it to the database.

2.4 Interactions

This module will be triggered by user events such as clicking the "Register" button. The module will store information in the database in the "account" and "token" tables. The module will also send an email to the email account that the user registered with.

2.5 Processing

The user will go to the registration page of the website. The user will fill out a form with the necessary information detailed below in section 2.6 and will click the "Register" button. This will trigger a script that checks if the input data is valid or not. If it is, the script will generate a random salt value and hash the user's password with it. The script will then attempt to insert the new account into the "account" database table. If an exception occurs that indicates that the entered email already exists as an account, an error message will be relayed back to the user. If the account does not exist already, the user's input will be stored in the "account" table and a token will be generated in the "token" table. This token will exist as a way to track if the user has verified their account. The script will generate a unique token value and random salt value that will hash the token value. An email will be sent to the email account that the user entered. It will contain a unique link that the user can open to verify their account that contains their plaintext token value. When they open the link, their plaintext token value will be hashed and compared to the hashed token value in the "token" table. If it matches, another script will run that will enable their account to log in.

2.6 Data

The following data is based on user input and will be stored in the "account" table:

Variable	Description
first_name	The user's first name
last_name	The user's last name
email	The user's email account
password_hash	The hashed version of the user's password
expected_graduation_date	The user's expected graduation date
track	The user's education track

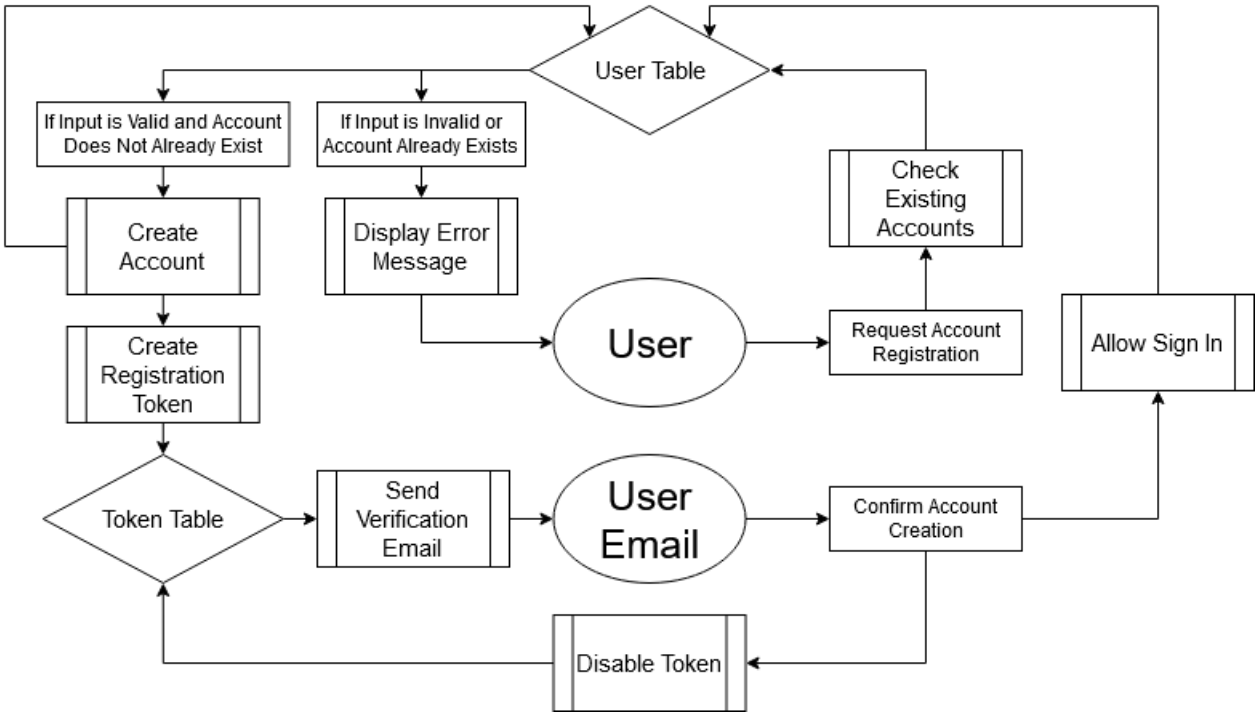
The following data is created entirely by the registration module and will be stored in the "account" table:

Variable	Description	Additional Information
id	The ID used to identify the user's account	
salt	The unique salt value that is used to hash a user's password	
account_type	The type of the account that the user has (normal user, moderator, admin)	Default value is "u" (normal user)
email_verified	Tracks if the user has verified their account	Default value is FALSE

The following data is created entirely by the registration module and will be stored in the "token" table:

Variable	Description
token_hash	The hashed version of the token value
salt	The unique salt value that is used to hash the token's value
account	The ID used to identify the user's account
expiration_time	The time that the token expires
is_valid	Tracks if the token has been used, has expired, or is able to be used
type	The type of token indicating what it is used for (password reset, account creation, etc.)

2.7 Relevant Diagrams for Module 1



3 Module 2 - Password Reset

3.1 Classification

Password Reset is a module consisting of multiple functions that will allow an existing user in the system to change their password without sending an email to the administrator asking "can you change my passowrd to this" or "what was my password again?".

3.2 Purpose

This module provides the functionality needed for a user in the system to change their password on their own.

3.3 Constraints

The process should be quick and easy for even the least skilled of users. The system should quickly get through the steps to send a verification email, and the processing of the password change should also be quick so that the user can log in as soon as possible using their new password. The process should only be initiated by a user, and should have safety measures set up to prevent SQL injections and other data entry-based attacks. The information will be processed to prevent these kinds of vulnerabilities before it is encrypted and forwarded to the database, where it will be stored.

3.4 Interactions

This module will not be used by any units, as it is the culmination of many units working together. These units that this module will use are "Check Existing Accounts", "Display Error Message" if the account does not exist, "Create Password Change Token" if the account exists, and puts it into the Token Table. It will also use "Send Verification Email" and finally "Change Password."

3.5 Processing

The user will go to the login page of the website. The user will then select "Reset Password" and be taken to a simple page. The user provide their email and will click the "Submit" button. This will trigger a script that checks if the input data is valid or not. If it is, the system will check existing accounts in the user table to see if the one submitted exists. If an exception occurs from the email given by the user not existing in the system, an error message will be relayed back to the user. If the account already exists, a token will be generated in the Token Table. This token will exist as a way to authenticate the identity of the user and authorize them to change their password. The script will generate a unique token value and random salt value that will hash the token value. An email will be sent to the email account that the user entered. It will contain a unique link that the user can open to change their password that contains their plaintext token value. When they open the link, their plaintext token value will be hashed and compared to the hashed token value in the "token" table. If it matches, another script will run that will enable the user to change their password. There will be two fields for enter password and confirm password. Once those are submitted by the user, a script will determine if the two match. If there is an exception, an error message will be displayed and prompt the user to try again. If they match, the new password will replace the old one in the User Table.

3.6 Data

The following data is based on user input and will be stored in the Account Table:

Variable	Description
password_hash	The hashed version of the user's password

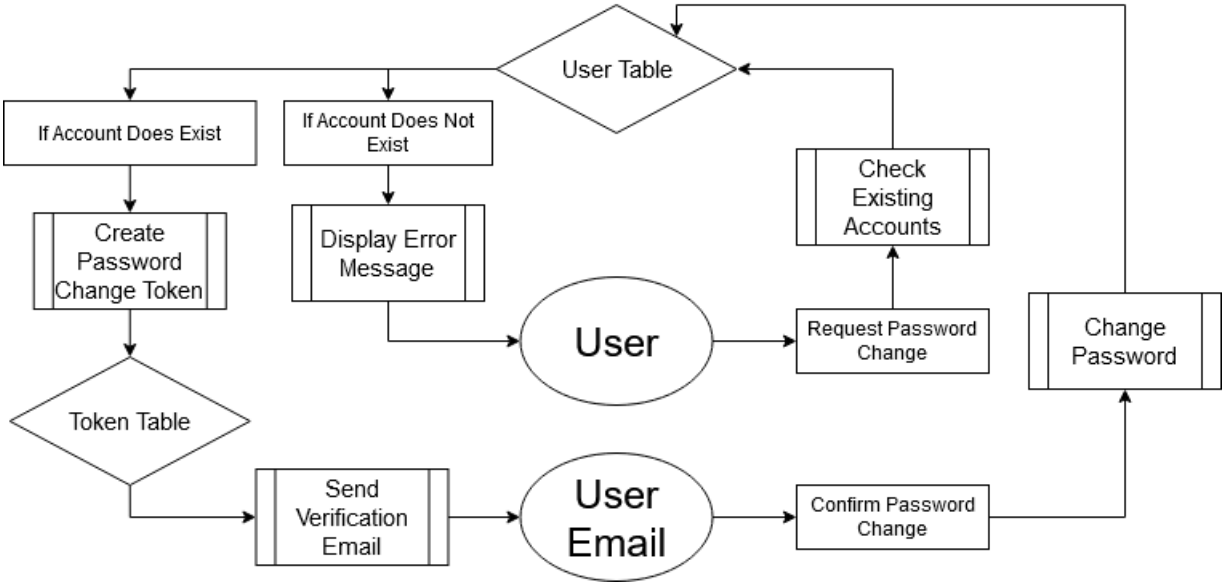
The following data is created entirely by the Password Reset module and will be stored in the Account Table:

Variable	Description
salt	The unique salt value that is used to hash a user's password

The following data is created entirely by the Password Reset module and will be stored in the Token Table:

Variable	Description
token_hash	The hashed version of the token value
salt	The unique salt value that is used to hash the token's value
account	The ID used to identify the user's account
expiration_time	The time that the token expires
is_valid	Tracks if the token has been used, has expired, or is able to be used
type	The type of token indicating what it is used for (password reset, account creation, etc.)

3.7 Relevant Diagrams for Module 2



4 Module 3 - Database Tables

4.1 Classification

Classification of the module (module, class, package, function, file)

4.2 Purpose

This component is the most important part of the whole product: the data.

4.3 Constraints

The data must be easy on storage. It must not be bloated with useless data. The data must be accessible: The users should be able to interact with the data quickly and easily, which relies on good database setup and good programming. The data must be confidential: the data will be precessed in a secure manner and whenever it is transferred, it will be done using secure methods. Much of the data will be hashed so that in the event of a breach, information of the useres remains safe. The data must also hhave integrity: there must be controls that moderate user interaction with data to prevent the data from being corrupted or misused.

4.4 Interactions

Practically everything will interact with the data through different functions. The users will interact with it through the software only for security.

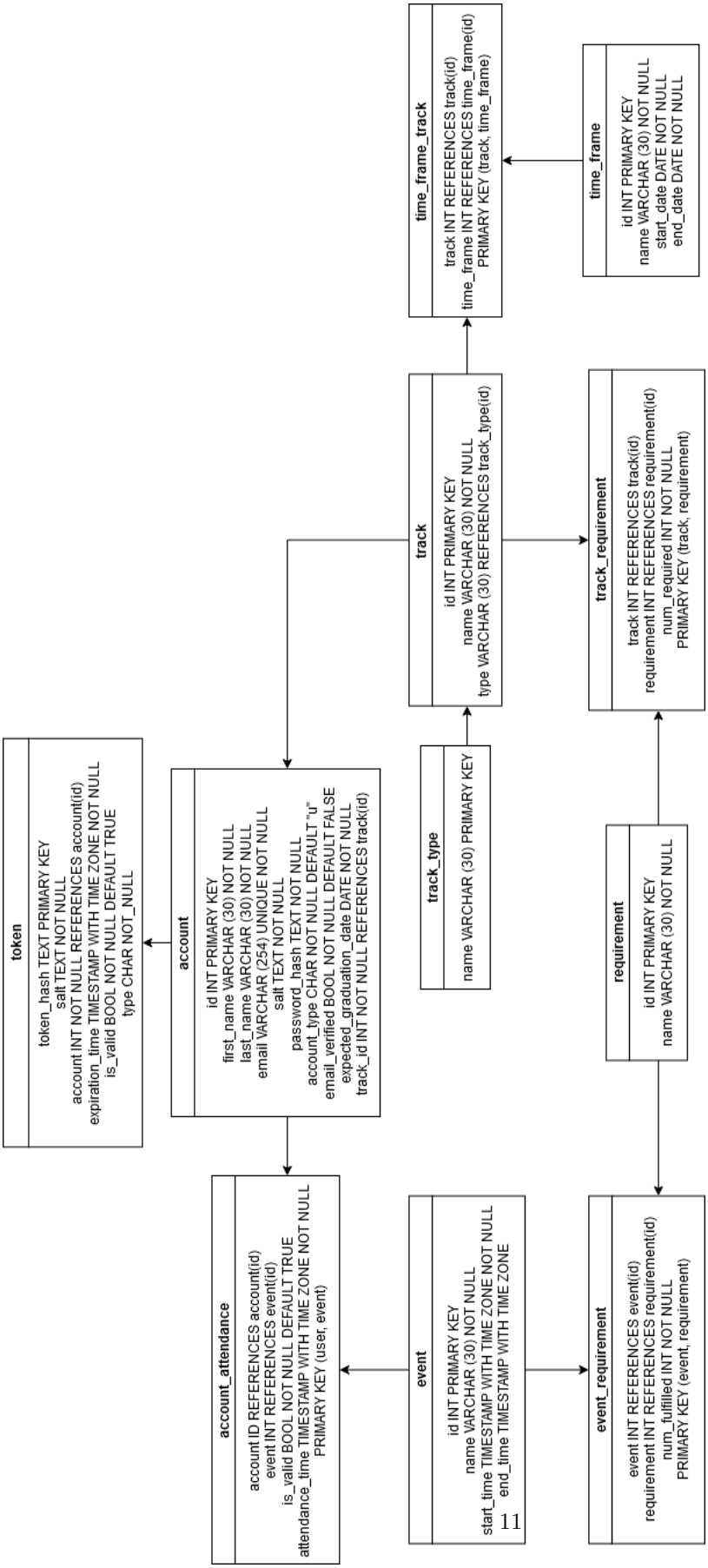
4.5 Processing

The different modules of the system will interact with the data. For instance, the Password Reset module will interface with the database and update data accordingly, while the user only interacts with the module. This paradigm allows the database to be accessed by no one, ensuring data security and error checking. The users will be allowed to interface only with the user facing modules.

4.6 Data

See the Relevant Diagrams for Module 3 subsection for the data types and their interactions.

4.7 Relevant Diagrams for Module 3



5 Module 4 - Get Attendance

5.1 Classification

Get Attendance is a module consisting of multiple functions that will allow an existing user in the system to view their attendance records or for a professor/admin user to view attendance records of all student users in a course.

5.2 Purpose

This module provides the functionality needed for a user to track attendance.

5.3 Constraints

The process should be quick and easy for even the least skilled of users. The system should quickly get through the steps to figure out what type of account is requesting the data, and provide the appropriate amount of data for that permission level. The process should only be initiated by a user.

5.4 Interactions

This module will not be used by any units, as it is the culmination of many units working together. These units that this module will use are "Get User's Account" if the account a student/normal-level account, "Get All User's Accounts" if the account is of type admin or moderator, and use those to fetch data from the AccountAttendance Table. It will also use "Display Attendance Records for Requested Account(s)" to display the information retrieved from the table.

5.5 Processing

The user will go to the attendance page of the website. This will trigger a script that checks what type of user is requesting the data from the AccountAttendance Table. If it is an admin or moderator account, the system will show the records or all accounts. If it is a normal student/user account, a script will run to get the information about the user requesting the data and will show the records for just that account.

5.6 Data

The following data is stored in the AccountAttendance Table:

Variable	Description
account	The foreign key to reference the corresponding user account.
event	The foreign key to reference the corresponding event.
is_valid	A boolean value to determine validity of the user being at the event.
attendance_time	A Primary Key DateTime variable to indicate the date and time that the attendance of the event was recorded.

5.7 Relevant Diagrams for Module 4