Keys To Win Academy

Prepared by

Lashawna Harris
Computer Science Major, Lewis University

CPSC 49200 Software Systems Capstone Project

Final Report, 8-27- 2022

Contents

Purpose

The goal of this project is to develop a kid-friendly web-based platform that will offer synchronous coding courses for disadvantaged children ages 8 – 11 years old. The platform will introduce children to foundational topics of computer science such as the meaning of linguistics, logical riddles, computational thinking, and coding games & challenges. The website will be bright, fun, alluring, and easy to navigate. User accounts will be created and stored in a database. Course content will also be stored in the database and assigned to individual users. Each child will be able to track their progression through the material unto completion. Parents will be able to request additional information to see if their student qualifies for the program.

Features

The proposed platform will offer the following features:

- Home Page
- About Us Page:
    - Provide mission and purpose
- Parent's Page:
    - Provide information about program
    - Link to form for interested parents/guardian
        - Registration Form:
            - Text fills for the following data:
                - Parent's/Guardian's first and last name
                - Parent's physical address
                - City
                - State
                - Zip Code
                - Parent's email address
                - Parent's phone number
                - Child(ren) names
                - Children date of birth
                - Gender – dropdown box
                - Ethnicity – dropdown box
                - Current grade

        - Request Information Form:
            - Text fills for the following data:
                - Parent's/Guardian's first and last name
                - Parent's email address
                - Parent's phone number
                - Child(ren) names
                - Children age

- Secure Login
    - Unique username and password for each child admitted
    - Parent unique login and password
- Course Content
    - PBS kids video tutorials & Crash Course tutorials
    - Various coding challenges from world-class organizations
    - Multiple Choice Quizzes at end of sections
- Course Content Progress Tracker
    - Use WordPress to create progress tracker
        - Tracker will automatically track when individual content is completed
        - Use bar or pie graph to show progress

User Interface

Home Page:

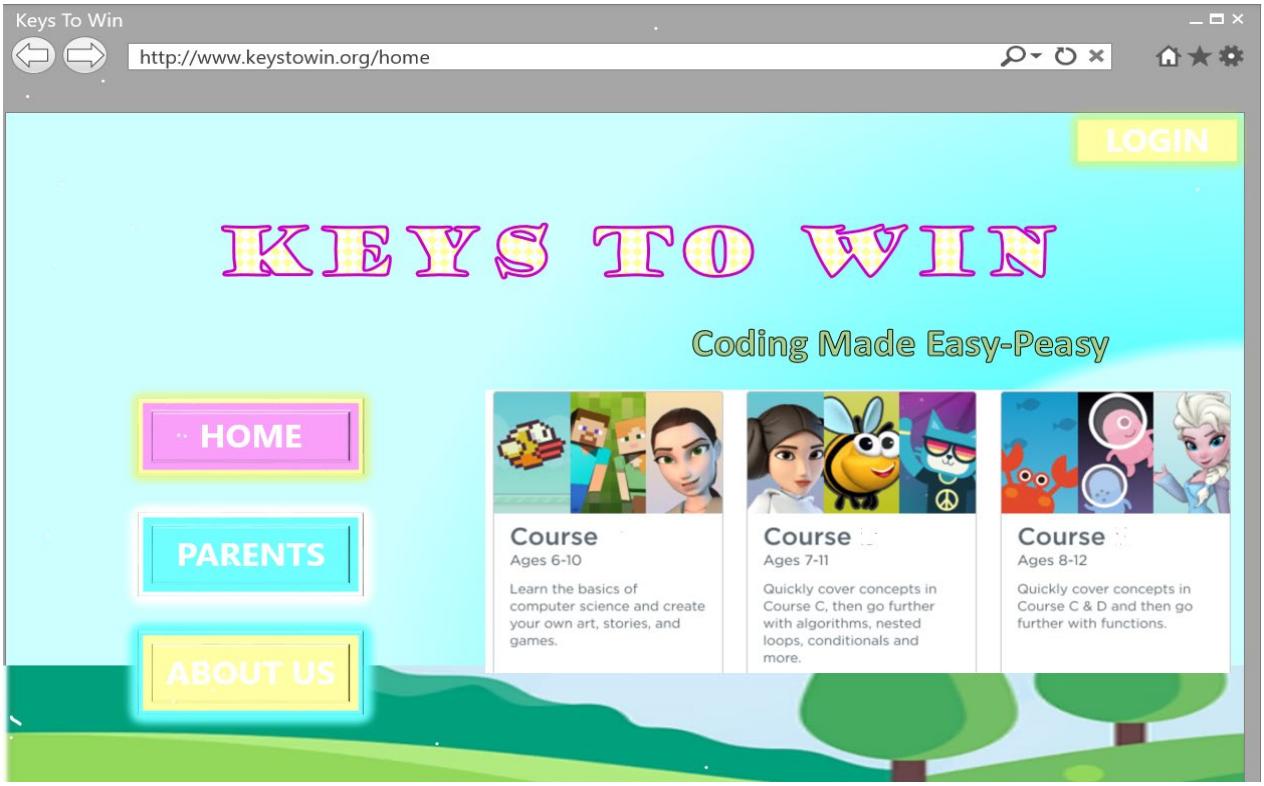


Figure 1 Mock-Up of Home Page of Proposed Website

About Us Page:

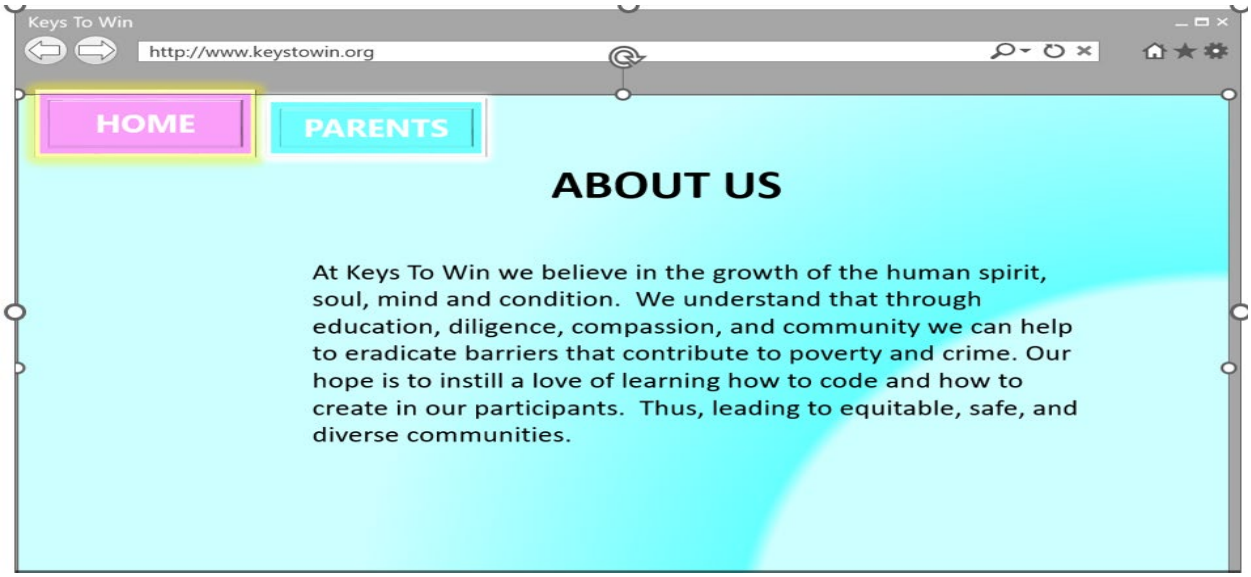


Figure 2: Mock-Up of About Us Page of Proposed Website

Parent's Page:

Figure 3: Mock-Up of Parent's Page of Proposed Website

More Information Request Form:

Figure 4: Mock-Up of Request More Information Form

Parent's Registration Page

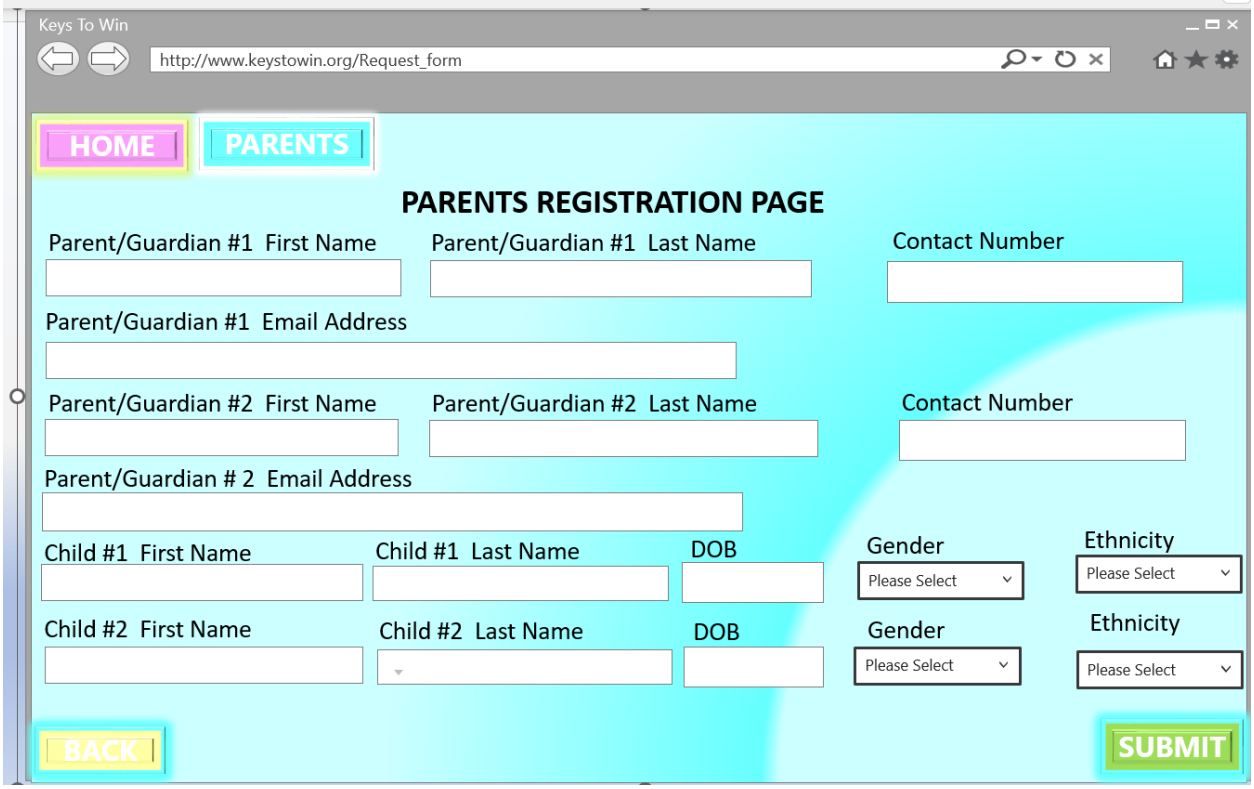


Figure 5: Mock-Up of Parent Registration Page

Alternatives

This project is designed as a digital platform for a non-profit initiative to increase the number of underrepresented students in computing.

--Code.org is spearheading free online coding material to students and schools. Code.org is advocating state-to-state for the inclusion of computer science as a required curriculum in primary schools. [1].

--GirlsWhoCode.com is another online coding platform that offers free coding content to future female engineers from ages 8-25. They plan on closing the gender gap for all new entry-level computing positions by 2030 [2].

--Future Ready is a City of Chicago Colleges post-pandemic initiative to get residences back to work through offering certifications that will provide new skillsets such as software development and entrepreneurship for free [3].

--Scratch.mit.edu is the world's largest free coding platform for children designed by Scratch Foundation of MIT. Their vision is to spread creative, caring, collaborative, equitable approaches to coding and learning around the world [4].

--Code with Google is a free platform created by Google to give every student a chance to explore, advance, and succeed in computer science [5].

Required Skills

Python	• Create functionality of platform (backend code logic)
SQLite	• To store user login credentials • To store course content such as videos, PowerPoints, etc.
Flask	• Utilize to provide server functionality for the platform
HTML / CSS/ Jinja	• Create structure and design of the User Interface
JavaScript	• Dynamically enhance the look and feel of the User Interface

Required Tools

Developing the website will require the following software

- PIP installed on local machine to install all necessary imports to Python
- Python 3 or higher to create the backend
- Flask to build out the server
- SQLAlchemy to be imported into Python to create database
- Spyder or Visual Studio Code for editing code on the local machine.
- PyTest import into Python for unit testing

Development Strategy and Timeline

The development of the application will be driven by milestones. The following table presents the milestones and their targeted completion dates.

Milestone	Target Completion Date
Proposal	May 31
Proposal Revision	June 6
Use Case Document	June 13
Class Design	June 27
Database Design	July 11
Connect Server, Code HTML and CSS/Unit Test	July 18
Write scripts for JavaScript/Unit Testing then Integration Testing	July 25
Load content into database/ Unit Testing then Integration Testing	August 1
Load content/create endpoints/ Unit Testing then Integration Testing	August 9
Experiential Learning Reflection/ Unit Testing then Integration Testing	August 16
Final Project Demo	August 23
Final Report	August 27

Testing

The testing of this application will happen in multiple sectors which will include extensive unit testing then integration testing for both frontend and backend code. I will utilize PyTest to perform the tests. The test iterations include but are not limited to:

- Ensure hyperlinks work for each web page on the website
- Ensure that user enter validate data for DOB, Phone#, Email
- Ensure data from request information form is sent to an email address for review

- Ensure that progress tracker considers all data for each section and shows a progress report
- Ensure video content is playing correctly
- Ensure that password requirements are implemented
- Ensure reset password works
- Ensure that unique login can only be created
- Enter user login will connect user to user specific content
- Ensure trouble reports are sent to admin for review

Licensing

This project will be licensed under Creative Commons. This license will provide the protection and freedom of Attribution-Non Commercial-Share Alike content. What this means is the following:

1. The content will be shareable
    a. The content can be copied and redistributed under any medium
2. The content will be adaptable
    a. The content can be built upon, remixed, and transformed as long as alterations are distributed under original license.
3. Attribution must be given— Appropriate credit must be given with a link to the license provided
4. Material may not be used for commercial gain
5. No new restrictions can be added to the original license

Use Cases

User Logs in.

To log into the system the user will click the login button on the top right corner of the Home Page. Next, the Login button will take the user to a login screen. This screen will prompt the user for the username and password.

If the user successfully logs in the system will take the user to a welcome screen that has a snapshot dashboard in the center of the page, a sidebar menu on the left that provides links to user specific reports and/or content. The page also will have a footer and logout button.

If the user does not login in successfully, the user will be alerted by a message in red letters staying "Login Unsuccessful. Please try again". The user also will have a "Forgot Password" link to reset password. Once password has been reset, user will be able to login.

User Logs Out.

Once logged into the system, in the top right corner there will be a log out button. When the user clicks the

Log out button the screen will be cleared

And a message will be provided **"You have**

Successfully logged out" .

User Logs Out.

Once logged into the system, in the top right corner there will be a log out button. When the user clicks the

Log out button the screen will be cleared

And a message will be provided **“You have Successfully logged out”** .

User Forgot Password.

If the user is having difficulty logging in the user can click on the “Forgot Password” link on the Login page to reset their password. The link will take the user to a “Recover Password” web page that will have an input field that request the email address associated with the account. The dialog box will also have a submit request button and a link to the login page just in case the user remembers the password. .

- **User Receives Confirmation Email.**

The user will receive an email that has a link with a token to reset the password. The reset password link will lead to a dialog box that will have a password field that will prompt for a new password and a confirm password field to confirm the new password. Once the password is successfully changed, the user will receive a message on the screen advising "**Your change has been successful**" and a link to the login page.

- **User Forgot Password.**

If the user is having difficulty logging in the user can click on the "Forgot Password" link on the Login page to reset their password. The link will take the user to a "Recover Password" web page that will have an input field that request the email address associated with the account. The dialog box will also have a submit request button and a link to the login page just in case the user remembers the password. .

- **User Change Password.**

The reset password link will lead to a web page that will have a password field that will prompt for a new password and a confirm password field to confirm the new password. Once the password is successfully changed, the user will receive a message on the screen advising "**Your change has been successful**" and a link to the login page.

- **Visitor view Home page.**
- When a visitor types in www.keystowin.org the home page for the platform will appear. On the home page in the main section there will be a header with the title of the platform. There also will be sample video content of a few courses. In addition, there will be a sidebar on the left that will provide links to the Parents Corner and About Us page. In the top right corner will be a login button that will take users to a login page.

Visitor view About Us page.

From the Home page a user can click the “About Us” link in the left sidebar to access the About Us page. The About Us page will have a header, footer, and left sidebar that offers links to the Home page and Parents’ Corner. This page will provide the mission and purpose of the organization.

- **Visitor view Home page.**
- When a visitor types in www.keystowin.org the home page for the platform will appear. On the home page in the main section there will be a header with the title of the platform. There also will be sample video content of a few courses. In addition, there will be a sidebar on the left that will provide links to the Parents Corner and About Us page. In the top right corner will be a login button that will take users to a login page.

Visitor view Parents' Corner page.

From the Home page a user can click the Parents link in the left sidebar to access Parents' Corner. Parents' Corner will have a header, footer, and left sidebar that offers links to the Home page, Request More Information, and an About Us page. Parents' Corner will provide statical data on how learning computer science is helpful for students. This page will also provide requirements for the program. It also will provide links to third party content.

- **Visitor view Home page.**

When a visitor types in www.keystowin.org the home page for the platform will appear. On the home page in the main section there will be a header with the title of the platform. There also will be sample video content of a few courses. In addition, there will be a sidebar on the left that will provide links to the Parents Corner and About Us page. In the top right corner will be a login button that will take users to a login page.

Visitor request more information.

Parents of potential students can request more information by clicking the “Request more information” link that will be on the sidebar in the Parents’ Corner page. This link will open up to a form that will require the parent’s and children first and last names, parent’s email address, parent’s phone number, and children’s age. The form will be sent to an email box that the administrator will monitor and respond to.

Parent register account.

Once the parent has decided to enroll their student. On the Parents’ Corner page, there will be a registration button where parents can register a family account. Once the button is clicked it will redirect to a registration form that will ask for parent information as will as the student(s) information. Once the correct information is submitted the data will flow over to a database and store the parents’ personal data in the parent’s table while, storing the children’s data in the children’s table.

Parent create login credentials.

Once the parent's account is registered the parent will be redirected to a form that will have a field to create a username, a password, and confirm password for the parent. The username and password must each meet certain length and complexity criteria. If the criteria is met the parent will receive a message stating the username and password has been created. If the criteria is not met the parent will be prompted in red fine print to fulfill the login credential requirements.

Parent views parent's dashboard.

The parent dashboard will have a header, footer, and main content section. In the main content section, the parent will be able to toggle between each child's record in order to view the individual child's progress

Student views dashboard.

Once the student is logged in the page will redirect to the student's dashboard. On the dashboard page there will be a header, footer, main content section, and left sidebar. The main content page will show the student's progress thus far. On the left sidebar, "Go to Course" link will be there so the student can enter the course learning path. There will also be a link on the sidebar for the student to contact the program administrator to request help.

Student go to course.

From the student dashboard page, the student will select the link "Go to Course" from the left sidebar. This will redirect the page to the student's learning path. The learning path will have all available course work for the student. All courses work completed will show a green check mark and state "completed". Course content left to complete will show as "to be completed" in chronological order.

Student clicks screen reader.

In the student's learning path, the student can click the play button on the screen reader so that the screen reader will read the text of content that will be on a slide. The screen reader will automatically discontinue once it reaches the end of the text or the end of the page, whichever comes first.

Student watches tutorial.

In the student's learning path, multiple videos will be in the list of items to be completed. Once the learning path is at a video, the student will click the play button on the video to watch the tutorial. The tutorial will be streamed from the database on the server to the UI. Once the tutorial is completed the system will automatically check it as completed.

Student watches tutorial.

In the student's learning path, multiple videos will be in the list of items to be completed. Once the learning path is at a video, the student will click the play button on the video to watch the tutorial. The tutorial will be streamed from the database on the server to the UI. Once the tutorial is completed the system will automatically check it as completed.

Student progress through learning path.

On the dashboard the student will be shown where they are at in the course learning path. If they are just starting the learning path they will start with a welcome video and written instructions. Once the student clicks the play icon on the welcome video it will play. From there the content will either auto play or the student will be prompted to move forward from lesson to lesson within the collection of material until the list of content is completed. If student has already begun the learning path the student will be able to pick up at the marker in which the student stopped.

Student receives certificate of completion.

Once the student has successfully navigated through the learning path and has completed all associated lessons within the collection the system will generate a certificate of completion that can be printed.

Administrator Search for student.

On the dashboard the Administrator will be able to search the database for student using a combo box that will provide the options of:

- Currently Enrolled
- Course Completed
- Student's Last Name
- student's First Name
- Parent's Last Name
- Enrollment Year
- All Students
- Students Completed

Once an option is selected the admin will input the required text. If the data is available a list will be provided with links to each individual entry. If the inquiry does not match an entry in the database a error message will occur stating "information not found"

Administrator Search for courses.

On the dashboard the Administrator will be able to search the database for courses using a search field. The admin will be able to search by course name or course number. The system will check the query against the database records. If the information is found, a link to the course content will be returned. Else an error message will display “no course found”.

Database Design

SQLite3:

I chose SQLite3 because Python has a built-in SQLite3 module. The module will be imported to interact with the database. Data in SQLite is stored in tables and columns, so you first I will create a table called `posts` with the necessary columns. I will create a `.sql` file that contains SQL commands to create the `posts` table with a few columns. I will then use this *schema file* to create the database. I also will create the Flask app files and template files then I will connect the environment on to a local server.

ERD Cardinality/Relationship:

1. Only one user can be logged in as the administrator.
2. The administrator can be different users, who login at different times.
3. Each administrator can print zero to many reports.
4. A report can be printed by any administrator.
5. One hashing algorithm id for one or many logins.
6. Zero to many users can receive an email validation letter.
7. Only one email validation letter can be sent per user login confirmation and recovery.
8. Each user can only have one login username.
9. Each username must be unique to the user.
10. There can only be one role per user.
11. Many users can have one role.
12. One role may be granted permission.
13. There could be zero to many granted permissions per role.
14. One permission can be given at a time to many roles requesting if qualify.
15. There can be no user that is a student.
16. There can be zero to many students that are users.
17. A parent can be a user.
18. There can be many users that are parents.
19. A student can only have one and only one parent.
20. A parent can have zero to many enrolled students.
21. Many students can have the same and only one parent enroll them.
22. A parent can have zero to many registered students.
23. Many students can have one and only one parent register them for courses.
24. Zero to many students can register for a course.
25. Each student can only be registered for one course at a time.
26. Each student can obtain zero to many milestones.
27. Many milestones can be obtained by only one student.
28. Each course offers zero to many milestone progress markers.
29. Each milestone is specific to each course.
30. Each course can have many lessons.
31. Every lesson is specific to one course.
32. Each course can have many quizzes.
33. Every quiz is specific to one course.
34. Each course can have many videos.
35. Every video is specific to one course.
36. Each section can have one to many courses.
37. Each course is specific to a section.
38. One to many students can be assigned to a section.

DB Tables

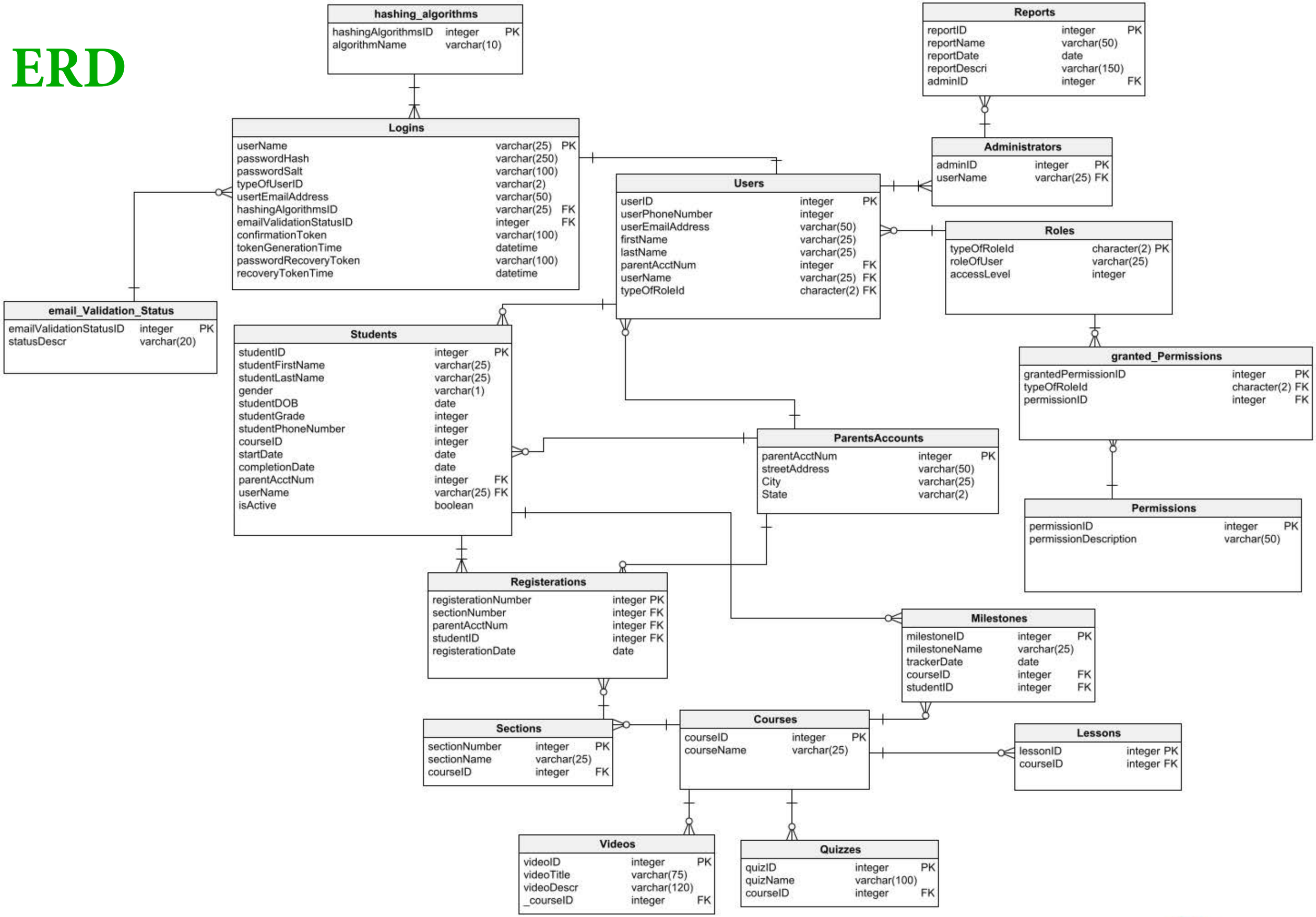
```
-- Table: Administrators
CREATE TABLE Administrators (
    adminID integer NOT NULL CONSTRAINT Administrators_pk PRIMARY KEY,
    userName varchar(25) NOT NULL,
    CONSTRAINT Administrators_Users FOREIGN KEY (userName)
    REFERENCES Users (userID)
);

-- Table: Courses
CREATE TABLE Courses (
    courseID integer NOT NULL CONSTRAINT Courses_pk PRIMARY KEY,
    courseName varchar(25) NOT NULL
);

-- Table: Lessons
CREATE TABLE Lessons (
    lessonID integer NOT NULL CONSTRAINT Lessons_pk PRIMARY KEY,
    courseID integer NOT NULL,
    CONSTRAINT Lessons_Courses FOREIGN KEY (courseID)
    REFERENCES Courses (courseID)
);

-- Table: Logins
CREATE TABLE Logins (
    userName varchar(25) NOT NULL CONSTRAINT Logins_pk PRIMARY KEY,
    passwordHash varchar(250) NOT NULL,
    passwordSalt varchar(100) NOT NULL,
    typeOfUserID varchar(2) NOT NULL,
    usertEmailAddress varchar(50) NOT NULL,
    hashingAlgorithmsID varchar(25) NOT NULL,
    emailValidationStatusID integer NOT NULL,
    confirmationToken varchar(100) NOT NULL,
    tokenGenerationTime datetime NOT NULL,
    passwordRecoveryToken varchar(100) NOT NULL,
    recoveryTokenTime datetime NOT NULL,
    CONSTRAINT Logins_hashing_algorithms FOREIGN KEY
(hashingAlgorithmsID)
    REFERENCES hashing_algorithms (hashingAlgorithmsID),
    CONSTRAINT Logins_email_Validation_Status FOREIGN KEY
(emailValidationStatusID)
    REFERENCES email_Validation_Status (emailValidationStatusID)
);

-- Table: Milestones
CREATE TABLE Milestones (
    milestoneID integer NOT NULL CONSTRAINT Milestones_pk PRIMARY KEY,
    milestoneName varchar(25) NOT NULL,
    trackerDate date NOT NULL,
    courseID integer NOT NULL,
    studentID integer NOT NULL,
    CONSTRAINT Paths_Courses FOREIGN KEY (courseID)
    REFERENCES Courses (courseID),
```

```
    CONSTRAINT Milestones_Students FOREIGN KEY (studentID)
    REFERENCES Students (studentID)
);

-- Table: ParentsAccounts
CREATE TABLE ParentsAccounts (
    parentAcctNum integer NOT NULL CONSTRAINT ParentsAccounts_pk
PRIMARY KEY,
    streetAddress varchar(50) NOT NULL,
    City varchar(25) NOT NULL,
    State varchar(2) NOT NULL,
    Zipcode character(5) NOT NULL
);

-- Table: Permissions
CREATE TABLE Permissions (
    permissionID integer NOT NULL CONSTRAINT Permissions_pk PRIMARY
KEY,
    permissionDescription varchar(50) NOT NULL
);

-- Table: Quizzes
CREATE TABLE Quizzes (
    quizID integer NOT NULL CONSTRAINT Quizzes_pk PRIMARY KEY,
    quizName varchar(100) NOT NULL,
    courseID integer NOT NULL,
    CONSTRAINT Quizzes_Courses FOREIGN KEY (courseID)
    REFERENCES Courses (courseID)
);

-- Table: Registerations
CREATE TABLE Registerations (
    registerationNumber integer NOT NULL CONSTRAINT Registerations_pk
PRIMARY KEY,
    sectionNumber integer NOT NULL,
    parentAcctNum integer NOT NULL,
    studentID integer NOT NULL,
    registerationDate date NOT NULL,
    CONSTRAINT Registerations_Sections FOREIGN KEY (sectionNumber)
    REFERENCES Sections (sectionNumber),
    CONSTRAINT Registerations_ParentsAccounts FOREIGN KEY
(parentAcctNum)
    REFERENCES ParentsAccounts (parentAcctNum),
    CONSTRAINT Registerations_Students FOREIGN KEY (studentID)
    REFERENCES Students (studentID)
);

-- Table: Reports
CREATE TABLE Reports (
    reportID integer NOT NULL CONSTRAINT Reports_pk PRIMARY KEY,
    reportName varchar(50) NOT NULL,
    reportDate date NOT NULL,
```

```
    reportDescri varchar(150) NOT NULL,
    adminID integer NOT NULL,
    CONSTRAINT Reports_Administrators FOREIGN KEY (adminID)
    REFERENCES Administrators (adminID)
);

-- Table: Roles
CREATE TABLE Roles (
    typeOfRoleId character(2) NOT NULL CONSTRAINT Roles_pk PRIMARY
KEY,
    roleOfUser varchar(25) NOT NULL,
    accessLevel integer NOT NULL
);

-- Table: Sections
CREATE TABLE Sections (
    sectionNumber integer NOT NULL CONSTRAINT Sections_pk PRIMARY KEY,
    sectionName varchar(25) NOT NULL,
    courseID integer NOT NULL,
    CONSTRAINT Sections_Courses FOREIGN KEY (courseID)
    REFERENCES Courses (courseID)
);

-- Table: Students
CREATE TABLE Students (
    studentID integer NOT NULL CONSTRAINT Students_pk PRIMARY KEY,
    studentFirstName varchar(25) NOT NULL,
    studentLastName varchar(25) NOT NULL,
    gender varchar(1) NOT NULL,
    studentDOB date NOT NULL,
    studentGrade integer NOT NULL,
    studentPhoneNumber integer NOT NULL,
    courseID integer NOT NULL,
    startDate date NOT NULL,
    completionDate date NOT NULL,
    parentAcctNum integer NOT NULL,
    userName varchar(25) NOT NULL,
    isActive boolean NOT NULL,
    CONSTRAINT Students_ParentsAccounts FOREIGN KEY (parentAcctNum)
    REFERENCES ParentsAccounts (parentAcctNum),
    CONSTRAINT Students_Users FOREIGN KEY (userName)
    REFERENCES Users (userID)
);

-- Table: Users
CREATE TABLE Users (
    userID integer NOT NULL CONSTRAINT Users_pk PRIMARY KEY,
    userPhoneNumber integer NOT NULL,
    userEmailAddress varchar(50) NOT NULL,
    firstName varchar(25) NOT NULL,
    lastName varchar(25) NOT NULL,
    parentAcctNum integer NOT NULL,
```

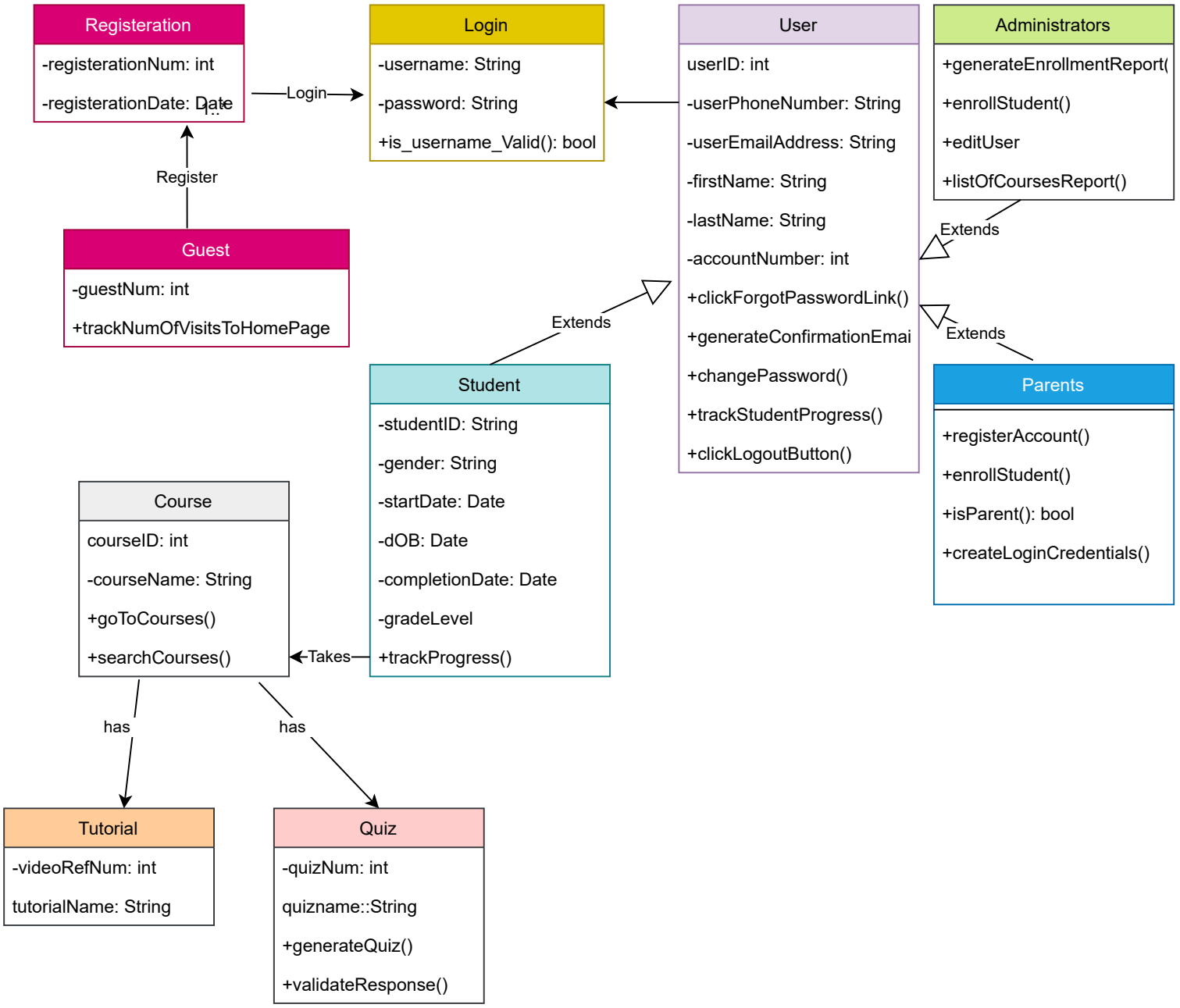
```
    userName varchar(25) NOT NULL,
    typeOfRoleId character(2) NOT NULL,
    CONSTRAINT Users_ParentsAccounts FOREIGN KEY (parentAcctNum)
    REFERENCES ParentsAccounts (parentAcctNum),
    CONSTRAINT Users_Logins FOREIGN KEY (userName)
    REFERENCES Logins (userName),
    CONSTRAINT Users_Roles FOREIGN KEY (typeOfRoleId)
    REFERENCES Roles (typeOfRoleId)
);

-- Table: Videos
CREATE TABLE Videos (
    videoID integer NOT NULL CONSTRAINT Videos_pk PRIMARY KEY,
    videoTitle varchar(75) NOT NULL,
    videoDescr varchar(120) NOT NULL,
    _courseID integer NOT NULL,
    CONSTRAINT Videos_Courses FOREIGN KEY (_courseID)
    REFERENCES Courses (courseID)
);

-- Table: email_Validation_Status
CREATE TABLE email_Validation_Status (
    emailValidationStatusID integer NOT NULL CONSTRAINT
email_Validation_Status_pk PRIMARY KEY,
    statusDescr varchar(20) NOT NULL
);

-- Table: granted_Permissions
CREATE TABLE granted_Permissions (
    grantedPermissionID integer NOT NULL CONSTRAINT
granted_Permissions_pk PRIMARY KEY,
    typeOfRoleId character(2) NOT NULL,
    permissionID integer NOT NULL,
    CONSTRAINT granted_Permissions_Roles FOREIGN KEY (typeOfRoleId)
    REFERENCES Roles (typeOfRoleId),
    CONSTRAINT granted_Permissions_Permissions FOREIGN KEY
(permissionID)
    REFERENCES Permissions (permissionID)
);

-- Table: hashing_algorithms
CREATE TABLE hashing_algorithms (
    hashingAlgorithmsID integer NOT NULL CONSTRAINT
hashing_algorithms_pk PRIMARY KEY,
    algorithmName varchar(10) NOT NULL
);

-- End of file.
```

ERD



Class Design

Controller	Function	Related Model	Related View
Users	login	User	user_login_view
	change_password	User	user_change_password_view
	register_account	User	retrieve_report_view
	password_reset_request	User	password_reset_request_view
	logout	User	no view necessary -reroute to home
Admin	is_active_student_report	Admin	active_student_view
	add_student	Admin	add_student_view
	admin_dashboard	Admin	admin_dashboard_view
	add_courses	Admin	add_course_view
	delete_user	Admin	delete_user_view
	update_course_description	Admin	update_course_descr_view
	retrieve_all_registered_users_report	Admin	retrieve_report_view
	track_student_progress	Admin	student_progress_view
Parent	enroll_student	Parent	enroll_student_view
	track_student_progress	Parent	track_student_progress_view
	send_msg_to_admin	Parent	send_msg_to_admin_view
	update_account_info	Parent	update_account_info_view
	delete_account	Parent	delete_account_view
	parent_dashboard	Parent	parent_dashboard_view
Student	student_dashboard	Student	student_dashboard_view
	go_to_course	Course	course_view
	request_certificate	Student	certification_view
Course	course_path	Course	course_view
	watch_tutorial	Course	course_view
Quiz			
	take_quiz	Quiz	quiz_view
	validate_answers	Quiz	quiz_view

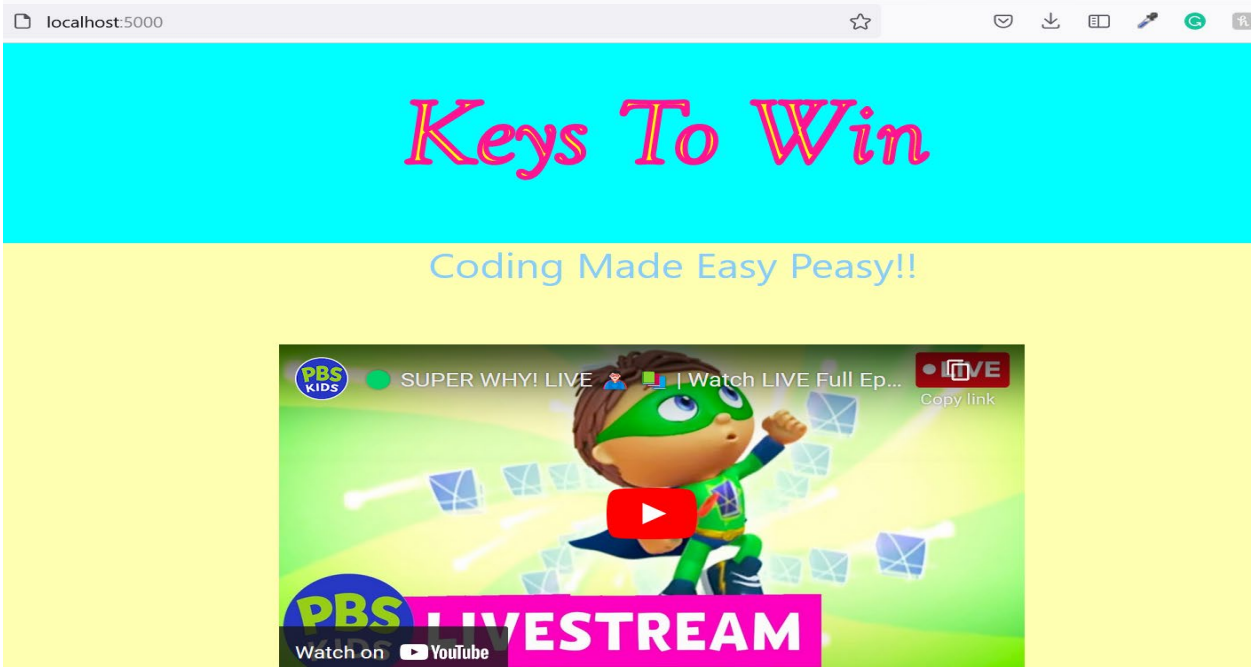

Registeration
-registerationNum: int
-registerationDate: Date
Login
-username: String
-password: String
+is_username_Valid(): bool
User
userID: int
-userPhoneNumber: String
-userEmailAddress: String
-firstName: String
-lastName: String
-accountNumber: int
+clickForgotPasswordLink()
+generateConfirmationEmai
+changePassword()
+trackStudentProgress()
+clickLogoutButton()
Administrators
+generateEnrollmentReport(
+enrollStudent()
+editUser
+listOfCoursesReport()
Login
Register
Guest
-guestNum: int
+trackNumOfVisitsToHomePage
Extends
Extends
Extends
Student
-studentID: String
-gender: String
-startDate: Date
-dOB: Date
-completionDate: Date
-gradeLevel
+trackProgress()
Parents
+registerAccount()
+enrollStudent()
+isParent(): bool
+createLoginCredentials()
Course
courseID: int
-courseName: String
+goToCourses()
+searchCourses()
Takes
has
has
Tutorial
-videoRefNum: int
tutorialName: String
Quiz
-quizNum: int
quizname::String
+generateQuiz()
+validateResponse()

Code Design

User Interface

Home Page – **Renders home page (homepage.html) to home page. Users will see this upon arriving to the site**

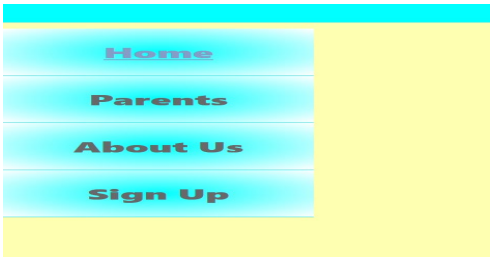
```
@views.route('/')
@views.route('/home')
def home():
    return render_template('homepage.html', user=current_user)
```



Other UI pages: **About Us Page gives our mission and vision. The Parent's Corner gives parents good to know information.**

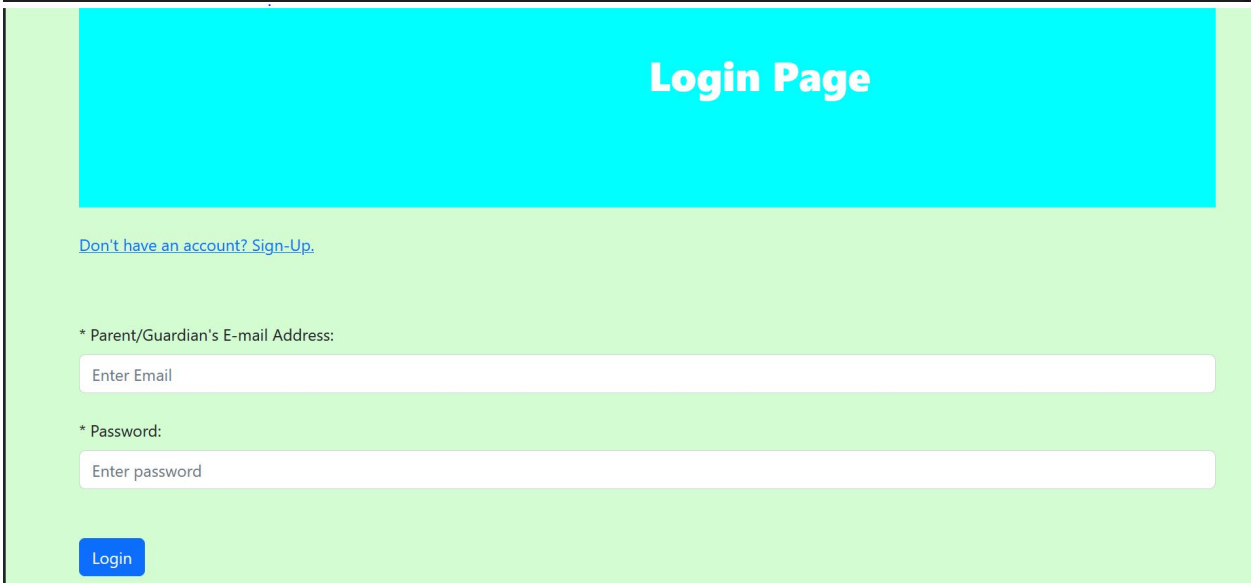
Left Side Navigation Bar – The navigation bar gives guests the options of clicking on a link to a different page on the site such as About Us Page, Parent's Corner, and Sign Up

```
</header>
<!--left side navigation bar-->
<nav>

    <ul>

        <li><b><a href="{{url_for('views.home')}}">Home</a></li>
        <li><a href="{{url_for('views.parents_page')}}">Parents </a></li>
        <li><a href="{{url_for('views.about_us')}}">About Us </a></li>
        <li><a href="{{url_for('auth.sign_up')}}">Sign Up</a></li>
        </b>
        </li>
    </ul>

</nav>
<main>

<div style="font-size: medium; color: darkorchid;"><br>
```



Login – Code that allows user to login by email and password. The database is queried first to confirm email exist, if the email exists then the password hash is checked against the data the user inputs. If the user's inputs are correct the user is redirected to the user dashboard. If the password is inaccurate a flash message is presented prompting the user to try again. If the email does not exist in the database, the guest is prompted to create an account.

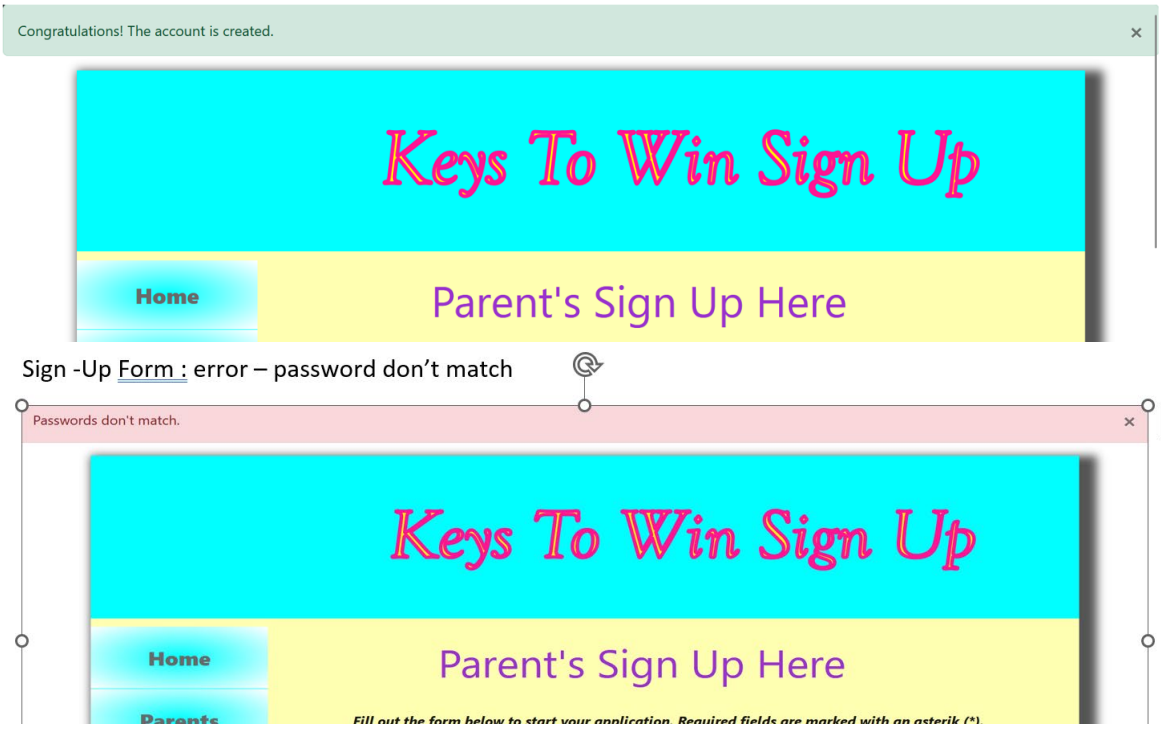
```python
@auth.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('myEmail')
        password = request.form.get('password')

        # query ensure email is valid email in db
        user1 = User.query.filter_by(email=email).first()
        if user1:
            # check if user enters correct password,  if so log user in - use hash for security

            if check_password_hash(user1.password, password):
                flash('Logged in successfully', category='success')
                login_user(user1, remember=True)
                return render_template('dashboard_student.html')
            else:
                # if password incorrect flash message to screen prompting user to try again
                flash('Incorrect password, try again.', category='error')
        else:
            # Email does not exist exist in db, for advise user to register an account.
            flash(
                'Email does not exist.  Check your input or  register an account.', category='error')

    return render_template('login.html', user1=current_user)
```



Sign Up Page - A parent utilize the sign up page to register an account. The parent's first name, last name, email address, password, and confirm password. Once the account is created the user information is stored in the database using 'POST' and the user is redirected to the login page to login.

```python
@auth.route('/sign_up', methods=['GET', 'POST'])
def sign_up():
    if request.method == 'POST':
        email = request.form.get('myEmail')
        first_name = request.form.get('firstName')
        last_name = request.form.get('lastName')
        password1 = request.form.get('password1')      You, 3 weeks ago • db created …
        password2 = request.form.get('password2')

        user = User.query.filter_by(email=email).first()
        if user:
            flash('Email already exists', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error')
        elif len(first_name) < 2:
            flash('First name must be greater than 1 character.', category='error')
        elif len(last_name) < 2:
            flash('Last name must be greater than 1 character.', category='error')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 7:
            flash('Password must be at least 7 characters.', category='error')
        else:
            new_user = User(email=email, first_name=first_name, last_name=last_name,
                            password=generate_password_hash(password1, method='sha256'))
            db.session.add(new_user)
            db.session.commit()
            login_user(new_user, remember=True)
            flash('Congratulations! The account is created.', category='success')
```

New Account Creation via form. When account creation is successful, user is automatically redirected to login page to sign in.



Sign -Up Form : error – password don't match

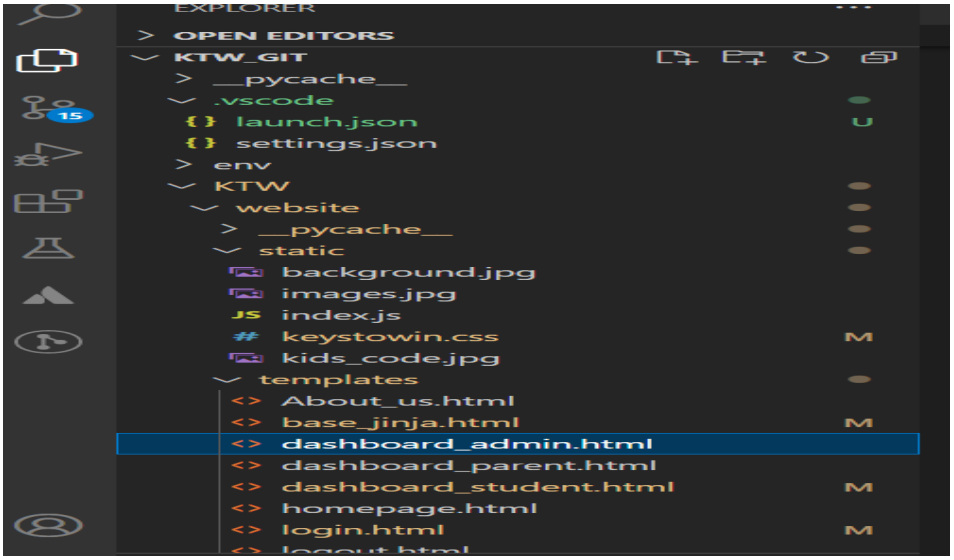
Login Manager – **built in Flask module that contains code that knows how to load a user from an ID, how to send each user to each user's personal account page. Each allows each individual login session to be tracked.**

```
# manage user logins
login_manager = LoginManager()
login_manager.login_view = 'auth.login'
login_manager.init_app(app)

# given user id return associated user
@login_manager.user_loader
def load_user(id):
    return User.query.get(str(id))      You, yesterday • update dashboards
```

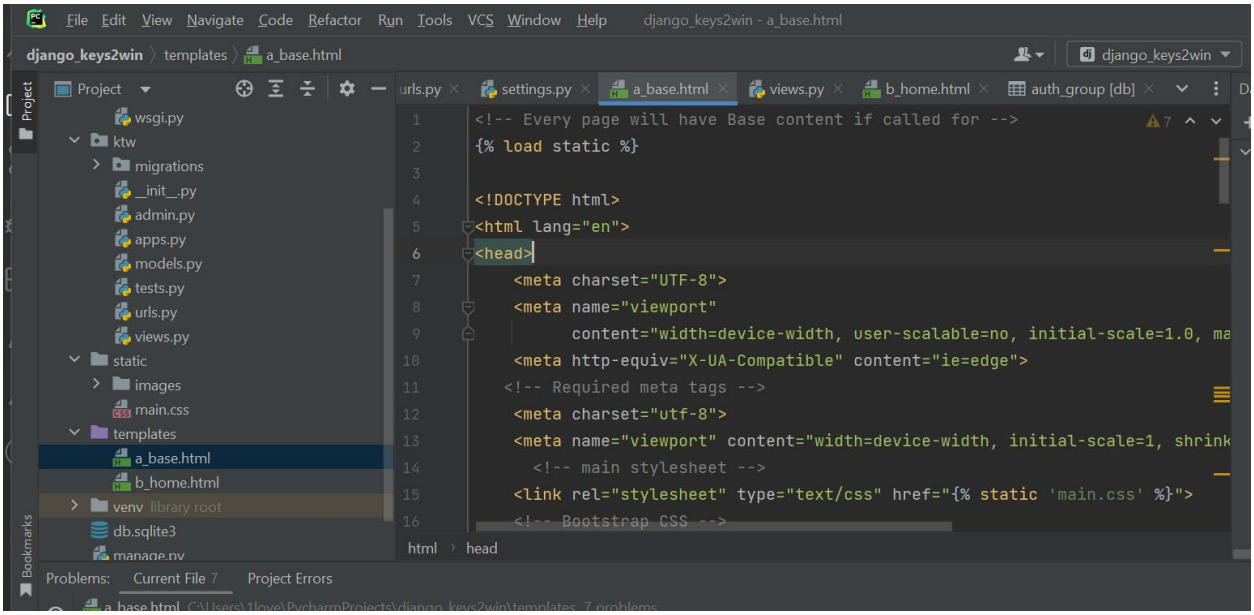
Blueprints- **Allows the developer to create a folder of files to be created into a module which can then be used throughout the application by importing the module into other components of the code base.**

```
# register blueprints
app.register_blueprint(views, url_prefix='/')
app.register_blueprint(auth, url_prefix='/')      Code Slayer, last
```

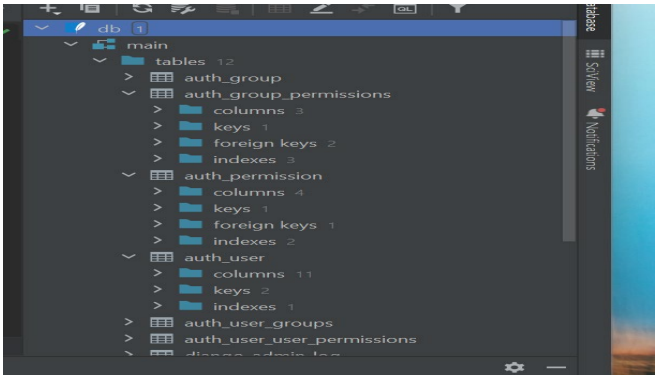
Snippet of Code Base – Code Base created in VSCode



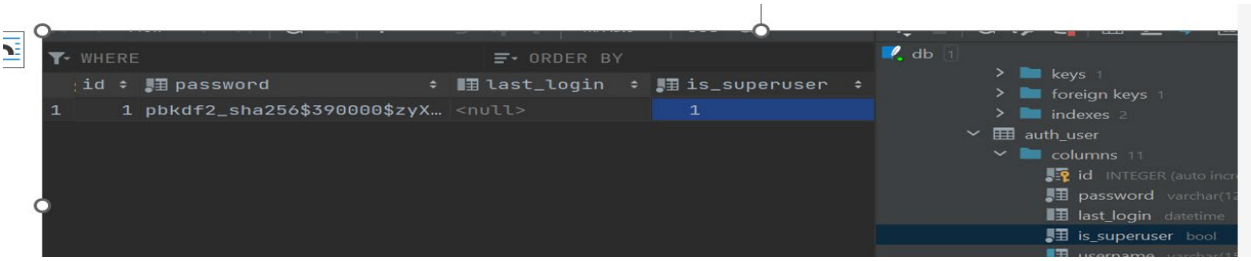
Code Base 2 – I began to build out code in Django using Pycharm with more organization --code shows bootstrap



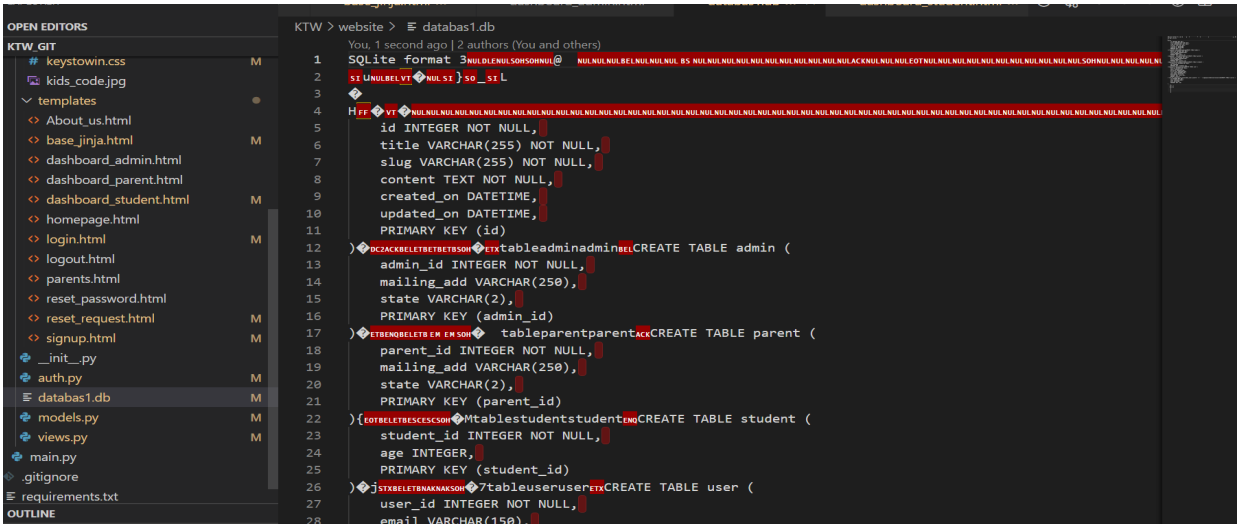
Database Migrated in Django – migrated database – most tables already built in with app



Admin created – superuser



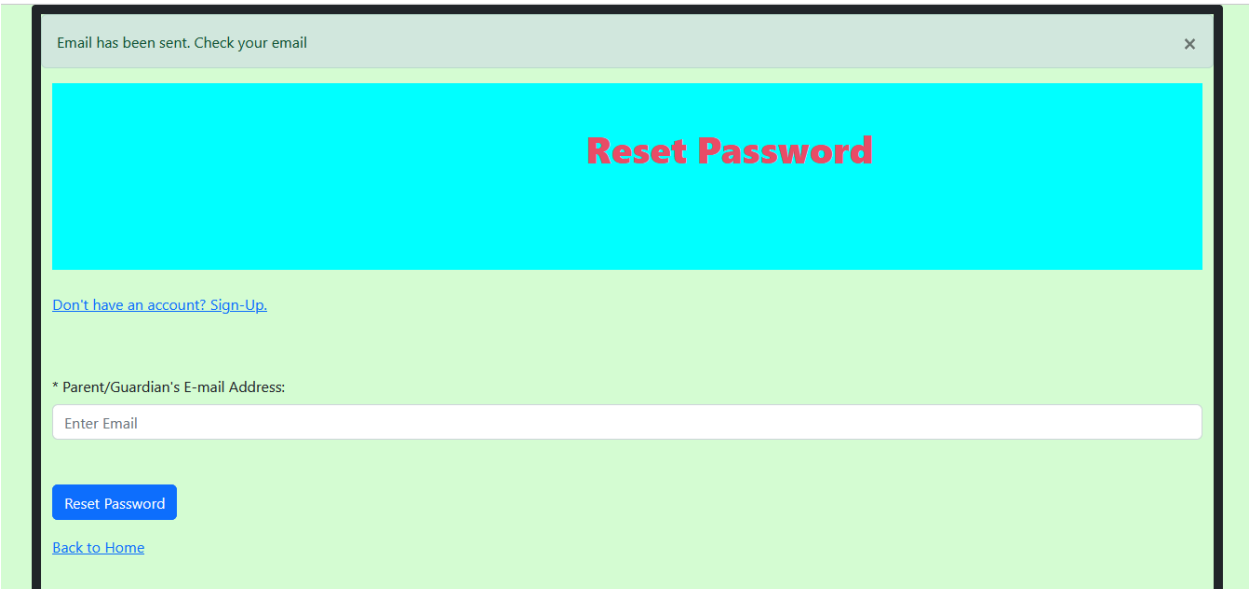
Database – VSCode



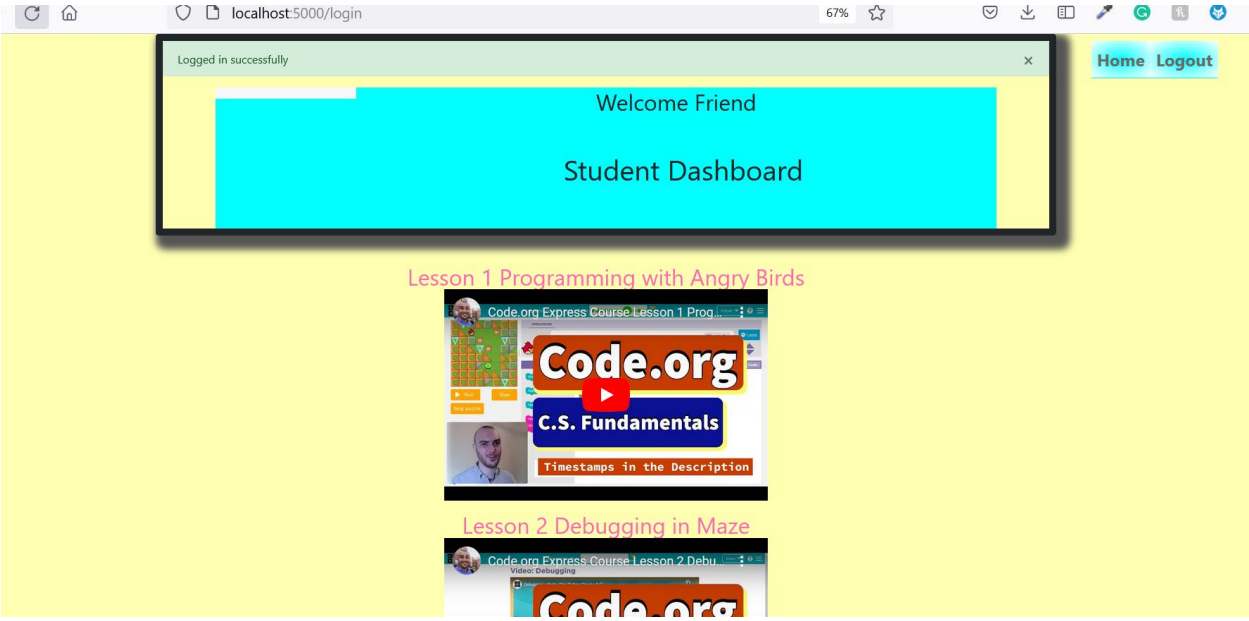
Forgot Password – Reset Request – **If a user forgets their password, while attempting to login a link is provided that will route the user to a reset request form. A message flashes advising users that an email will be sent to their email address.**

```python
@auth.route('/reset_request', methods=['GET', 'POST'])
def reset_request():

    email = request.form.get('myEmail')
    if email == email:
        flash('Email has been sent. Check your email', category='success')
    else:
        flash('Email not valid. Try again.', category='error')
    return render_template('reset_request.html', title='Reset Request')
```

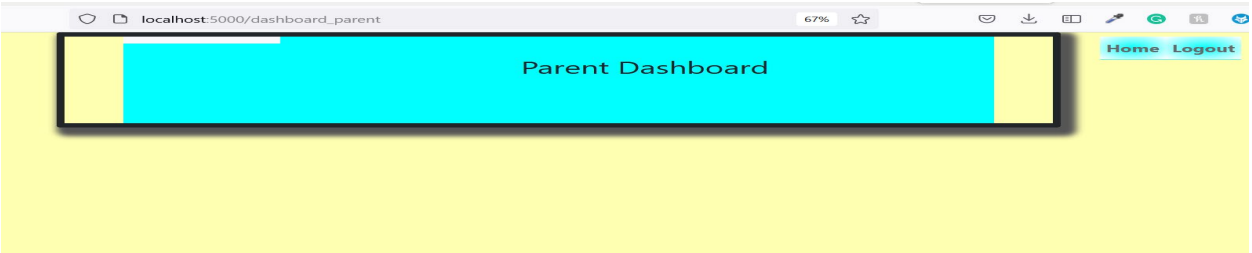
You, 5



User Logged In – When a user logs in the user is routed to the student's dashboard.



Parent's Dashboard - **When a parent login, the parent will be able to track their child's progress, enroll a child, and update their personal information.**



Admin Dashboard – Admin will have full CRUD capabilities. The admin will be able to generate reports, delete users, create users, update course descriptions, delete courses, add users, and add courses.

Test Plan

Keys to Win design has three different user types. The user types are:

- Admin
- Parent
- Student

Currently functionality for a user to login and be redirected to a view is created. The creation of user roles will be implemented in the future allowing for different test cases to be performed. So, for now the existing test cases will be performed for two sample “Users ” to demonstrate the routing to specific user profiles.

Sample 1: Existing user

Email address: cmayor@gmail.com Password: test123

- Once logged in user can begin “CS Fundamentals” learning path
    - Learning Path consist of multiple Code.Org videos. When user clicks on the video, the video will play.
- User can navigate to the Home page by clicking the Home button in the top left corner.
- User can logout.

Sample 2 User – email address and password of your choice!!

- Can visit Home page, About Us page, Parent’s Corner
- User can click on social media links at the bottom of the Home Page
- User can click on the Sign -Up link to be routed to the sign-up form
- User can sign-up using a valid email address
- User can log in. Once logged in, user can repeat steps of Sample User 1.

How To Access and Install Project

The Codebase for Keys To Win is publicly shared on GitHub.

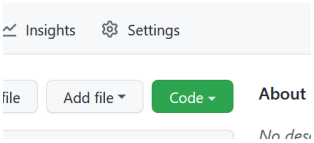
GitHub is a centralized hosting service for software development. It is currently the largest platform where millions of developers share code.

Step 1

To access my GitHub account please click this link, https://github.com/Shawna-Harris/ktw_git . This link takes you directly to the repository.

Step 2

At the top right, there is a green button labeled “Code” . Click on this button.



Step 3

Clone the repository. Cloning the repository will create a local version of the repository on your computer. GitHub provides a few different ways to download the project:

- Url (using HTTPS, SSH, or GitHib CLI)
- GitHub Desktop (if you have the desktop app installed on your pc)
- Download a zip file. Then extract to the desired directory on your pc.

If you will copy the URL, Choose GitHub CLI. Next, open Command Prompt. Once there change into the directory you would like to store the repo by typing: cd “directory path”. Now, make a new directory by typing “mkdir” for Windows, Linux, and Mac. Last, type git clone followed by the URL.

Step 4

Utilizing VsCode, PyCharm, Spyder, or the Integrated Development Environment (IDE) of your choice, open the directory folder that the repository is in.

Step 5

Navigate to Main.py and run the application.

Step 6

Optional – The ReadMe page provides all installed software.

Conclusion

Technology is integrated in every aspect of life. Thus, technical opportunities in computing are ample. Yet, due to the lack of training and skillset many Americans are continuing to feel the blow of the great Digital Divide. Keys To Win seeks to close the equity gap in hopes to create a more balanced society. Keys to Win has gone from a concept to a work in progress. As an artist, neighbor, and a friend I am very proud to be a part of this journey. Currently, the website is very appealing and friendly. My friendly and fun nature is expressed. However, the codebase is not a demonstration of my organizational skills. Therefore, it will be re-written. There will be many iterations of this project. The finished version will be located at 'keystowin.org'. There are many features that be implemented. Yet, I will list the next set of features that must be implemented. CRUD functionality must be extended to the following features :

- Retrieving Data, Updating Records, Deletion, Add
    - reports to show if a student is active, how many enrolled, all users, and student progress
    - parents will be able to update account profile; admin will be able to update course content description
    - Admin will be able to add new user, courses, videos, quizzes, reset passwords
    - parent will be able to enroll student, track student progress

References

[1] “Advocate for Computer Science Education,” *Code.org*. [Online]. Available: https://advocacy.code.org/. [Accessed: 31-May-2022].

[2] “About Us,” *Girls Who Code*, 14-Dec-2021. [Online]. Available: https://girlswhocode.com/about-us. [Accessed: 31-May-2022]. \

[3] “Future ready - short-term programs at no cost for eligible Chicagoans,” *City Colleges of Chicago: Apply Now*, 26-May-2022. [Online]. Available: https://pages.ccc.edu/apply/futureready/. [Accessed: 31-May-2022].

[4] “Our story,” *Scratch Foundation*. [Online]. Available: https://www.scratchfoundation.org/our-story. [Accessed: 31-May-2022].

[5] “Code with google | google for education,” *Google*. [Online]. Available: https://edu.google.com/intl/ALL_us/code-with-google/. [Accessed: 31-May-2022].

Appendix A: Resume

Summary resume

Lashawna:

Name	Lashawna Harris
Summary Sentence	A patient professional, with an outgoing attitude and problem-solving mentality
Address	1347 Crab Apple Ct #101, Naperville, IL 60540
Email	lashawnaharris@lewisu.edu
Phone	630-994-0317
Other contact info	
Education (College, Major, Minor, GPA)	Lewis University, Computer Science, 4.0 GPA
Software Development Skills	Java, Python, HTML/CSS
Information Technology Skills	Coding
Professional Skills	Communication, Teamwork, Flexibility, Time Management
Career Interests	Web development, Artificial Intelligence
Evidence of Work (e.g. portfolio, GitHub)	N/A
References	Susan Fenwick fenwicks@cod.edu , Dr. Eric Chou echou@lewisu.edu