Luna Summer 2017

University of Kentucky
CS485: Special Topics Class

Joshua (Dillon) Pulliam
dpu225@g.uky.edu

Danyse Hickman
drh223@g.uky.edu

Instructor:
Paul Piwowarski
paulp@cs.uky.edu

Sponsor/Client:
Kaylynne Glover
kaylynnemglover@gmail.com

View this document online at:
http://www.cs.uky.edu/~paulp/luna/index.html

Introduction

What is Luna?
Luna is an app that offers innovative features that allows a user to collect and store information about her menstrual cycle behavior. Luna's data will be used to collect research about fertility and dating behavior.
Using an existing database and server, our project is to create the Luna app the research participants will use to enter the data for the research study. The purpose of the app is to provide research into dating behaviors throughout a woman's menstrual cycle tracked by the app.

What will we do?

- Build the Luna app for research participants using the Ionic platform for iOS devices (iPhones, iPads)
- Update the user consent form and new user registration process
- Add user login email verification
- Update the database:
    - add on boarding questions entry columns to User table in the database
    - add daily questions columns to the Entry table that track participant's daily status

Requirements

New User Registration

New user accesses Luna host web page to register as research participant
New user must be between 18-25, and uses UK email address to get random ID to use with Luna app
New user access Luna app from Apple store, enters random ID sent to her email address, login user name and password (only used to access her personal data, not to daily questions)
Note: No identifying information can be attached to the research, app must maintain user's anonymity per IRB

On-boarding questions

Must all be answered before the user can access other features of the app
Information about the user (birthday, period length, etc.) important for the study
Saved in user table
Can be viewed and changed by user

Daily questions:

Daily questions record user's daily menstrual cycle information for this study
Recorded as new row in entry table for this user

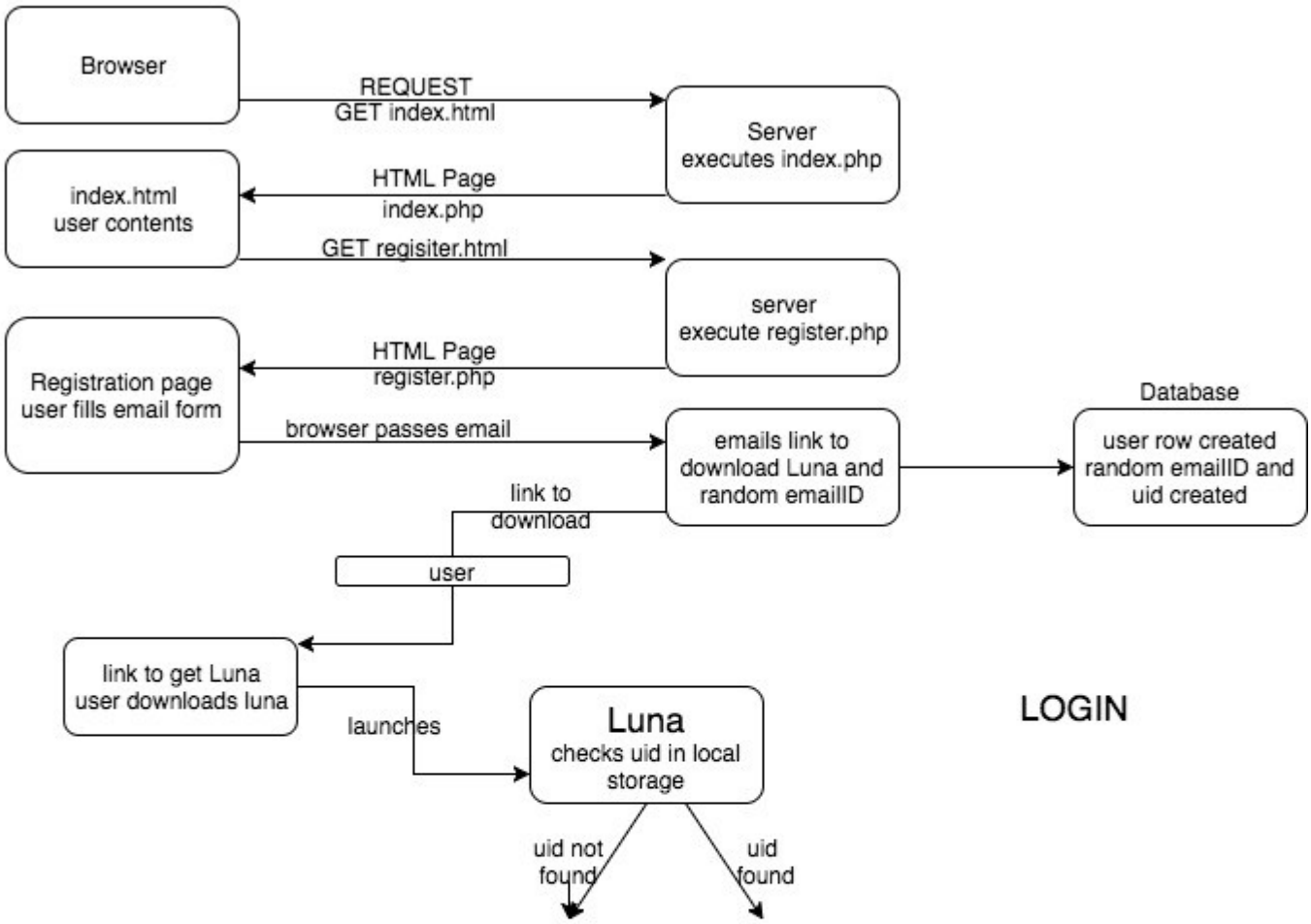
Main App Features

Luna has 3 tabs. These pages are as follow: tracker, settings, and calendar pages.

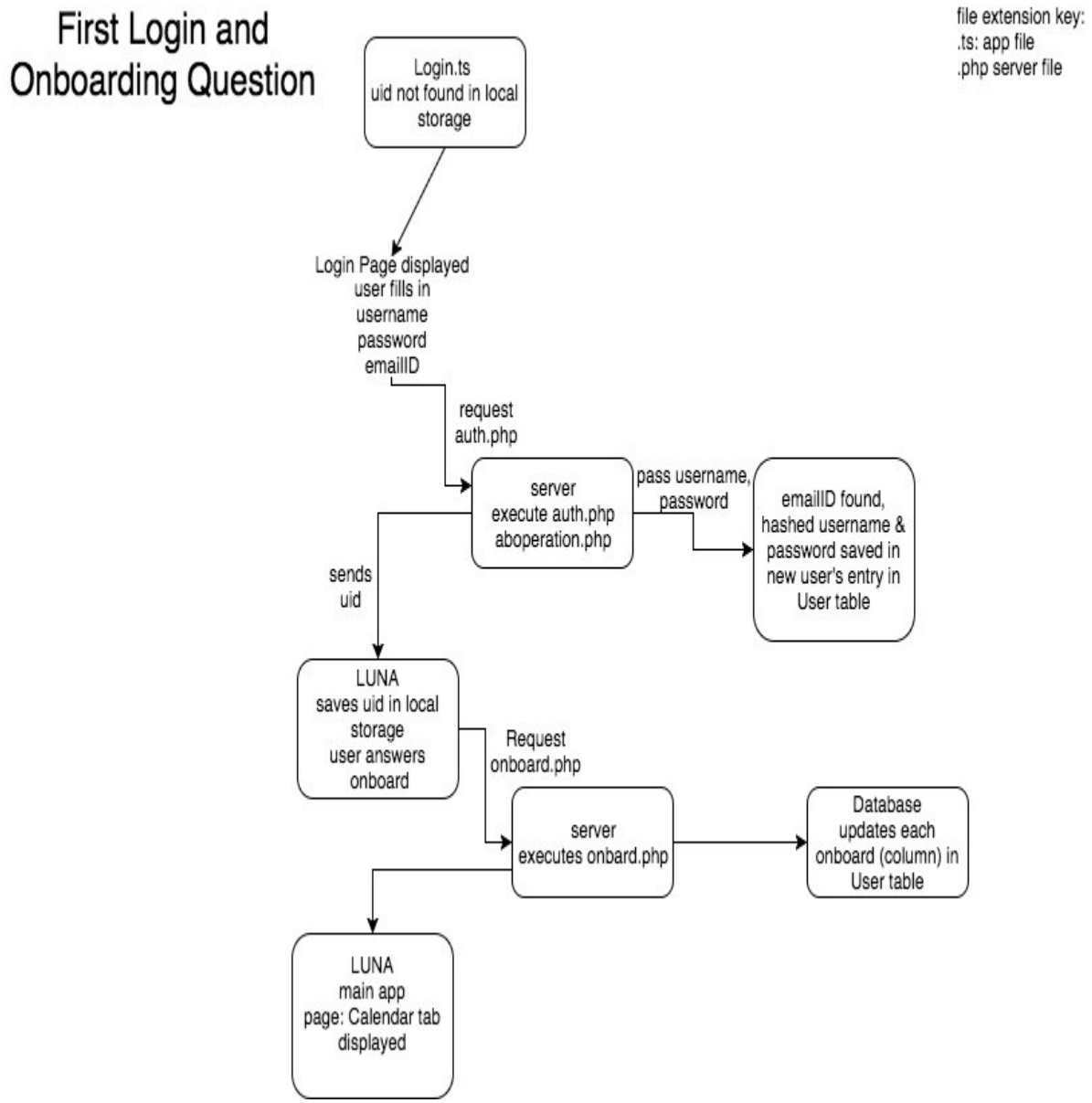
1. Tracker Page: Main tracker page used for user to supply answers to daily questions
2. Settings Page: Contains the onboard questions for users to update should their original answers change.
3. Calendar: Reminders for important dates for the user for the study

Luna Design

REGISTRATION



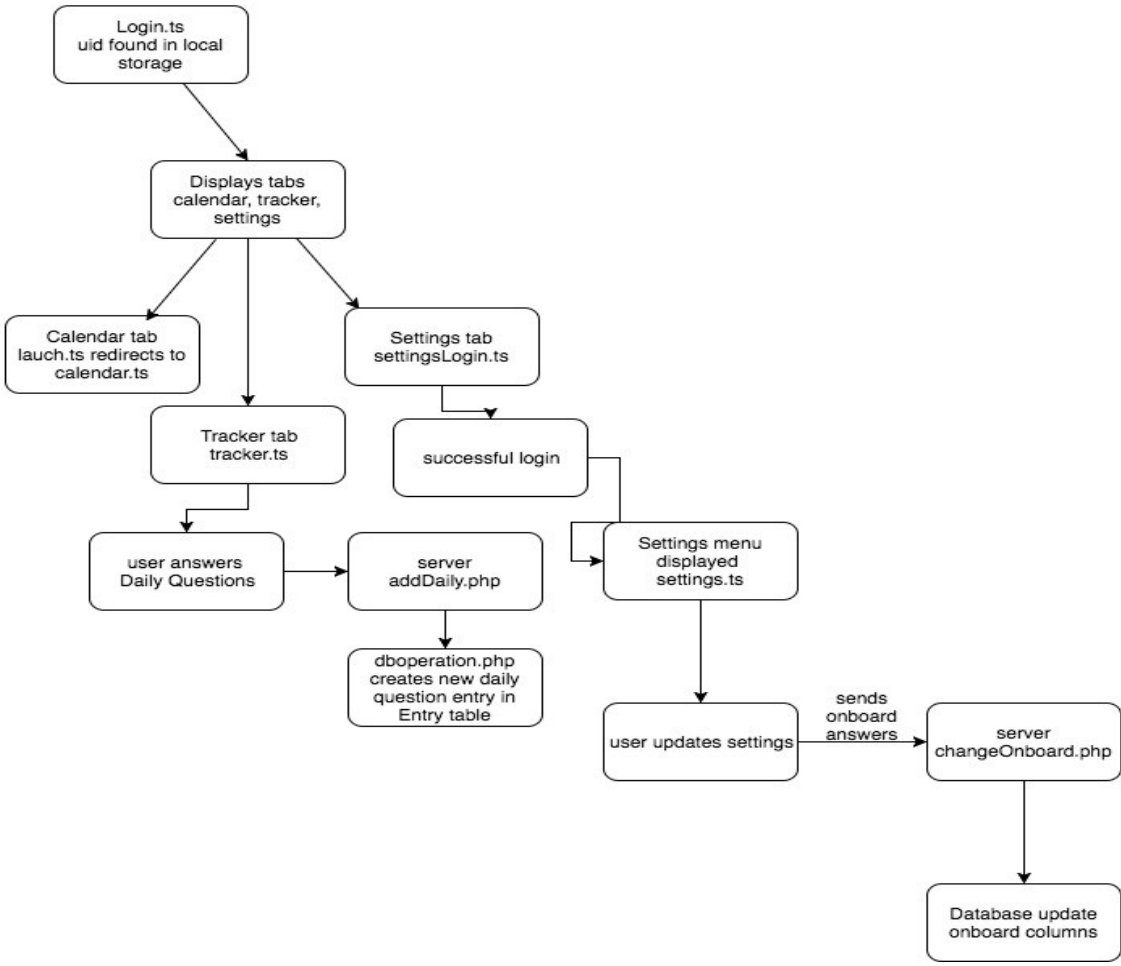
LOGIN


First Login and Onboarding Question
file extension key:
.ts: app file
.php server file
Login.ts
uid not found in local storage
Login Page displayed
user fills in
username
password
emailID
request
auth.php
server
execute auth.php
aboperation.php
pass username,
password
emailID found,
hashed username &
password saved in
new user's entry in
User table
sends
uid
LUNA
saves uid in local
storage
user answers
onboard
Request
onboard.php
server
executes onbard.php
Database
updates each
onboard (column) in
User table
LUNA
main app
page: Calendar tab
displayed

New User

file extension key:
.ts: app file
.php server file

Daily App Operation



Fields in the user table created during the new user process:

emailID: random string sent to new user email address (done in register.php)
username/password: supplied by new user in logon screen, stored in the user's row. Only used when user wants to access personal data in tables.
uid: user ID, random number created by register.php for new user, when new user row created in user table.

The uid is identifier that the app sends to the server for this user to change data in the tables.

1. New user uses browser to access Luna page: https://luna-app.000webhostapp.com
(executes index.php)
2. Consent form displayed, new user must agree. New user enters age (must be between 18-25)
3. New user fills in email address, presses submit. If UK email adderess, continue with step 4.
4. Server: register.php executed. User row added to user table, random emailID, random uid created, saved in new user table row
5. Random ID sent to email address (emailID column in user table)
6. Page displays instructions on how to get app
7. New user gets app, executes it on iPhone, or iPad
8. App checks if User ID (uid) has been saved in its local storage. If not, this is new user,

continue with step 9. If saved, displays main app page.
9. App displays logon page that asks for username, password, emailID
10. New user fills in emailID from email, username, password requested, presses submit
(sends request to server auth.php)
11. auth.php searches user table for emailID, if found continues
12. auth.php saves username, password in this new user's entry in user table, sets active flag
true
13. auth.php sends uid to app in message field of reply to app
14. app saves uid in local storage, displays page for new user to enter onBoard questions
15. After onboard questions answered, they are sent to server (onBoard.php), main app page

Onboard questions

1. The new user is directed to answer the onboard questions. The user can view or change the answers to onboard questions via the settings tab.
2. The user must supply answers to all of the onboard questions. If not, when submit is pressed, the an alert message is displayed with the name of the first answer not supplied.
3. When all onboard questions answered, and submit is pressed, the answers are sent to changeOnboard.php.

Daily questions

1. When user executes the app, it looks up the user's uid stored in the app's local storage.
2. If the local storage uid contains the user's uid, the main screen is displayed.
3. When the user presses the tracker tab, tracker.ts is executed. The first daily question is displayed.
4. The user answers questions d, and e (see the entry table structure in the Data Base Documentation below).
5. If the user responds positively to question f, the remaining daily questions are displyed.
5. f the user responds positively to question f, the remaining questions are displayed through question m.
6. Another toggle button is then displayed asking the user if they have had sexual stimulation or intercourse.
7. If the user responds to yes to this then the remaining questions are displayed through question p.
8. Another toggle button is then displayed asking the user if they have had intercourse.
9. If the user responds yes to this question then the remaining questions are shown.
10. When the user presses submit, the tracker.ts function sends the answers to addDaily.php in the server.
11. addDaily.php creates a new row in the entry table for this user's daily questions.

Settings

1. The user can display and change onboard questions and answers by pressing the settings
tab. The onBoarding.ts function is executed.
2. The user is asked for her username and password. If they match the variables in local
storage, the onboard questions are displayed.
3. The user can change any of them.

1. The settingsLogin.ts page is executed.
2. If username and password match then the settings.ts page is displayed which shows the responses to the onBoarding questions (saved in local storage) and allows a user to update their responses to some of these questions.
3. Only certain questions can be updated.
4. The app sends all the onboard questions to the server (changeOnboard.php). changeOnboard.php updates
the onboard columns in the user's row (identified by UID) in the user table.

Calendar

The calendar page is displayed at the end of the new user steps, and when the app is started for the daily questions. The user is prompted to press the tracker tab to answer the daily questions. Currently the calendar is not used to display any user information.

Database Documentation

Only the user table, and entry table are used by Luna. The period table and image table were created for possible future use. The incident table is used by the Sara app.

1. User Table

There is one row for each user. It is created during the "new user" activity. It is flagged active when the new user logs on for the first time. User rows are never deleted. The active flag can be set off, however this function is currently not provided.
a. uid - user identifier (primary key)
b. username - MD5 hashed username
c. password - MD5 hashed password
d. active - activity status (0 is inactive - can't log in)
e. onboard_status - onboarding status (1 is completed)
f. birthday: user's birthdate
g. cycleLength: the number of days from one period until the next
h. periodLength: the number of days menstruating
i. birthControlType: hormonal or non-hormonal
j. lastPeriod: date of user’s last menstruation
k. status: current relationship status, single, casual, married, and long-term
l. time: time of activity on Luna
m. pregnant: whether or not user is pregnant
n. reproductiveDisorder: Answers to onboarding questions- ‘have you been diagnosed with a reproductive disorder’

2. Entry Table

The entry table contains a row for each user's daily questions.

a. eid - entry identifier (primary key, auto increment)
b. uid - user identifier (foreign key)
c. entry_date - date of entry
d. onPeriod - - answer to daily question,menstruating status
e. sexualInterest - answer to daily question, rating of sexual desire or interest
f. sexualAcitvityNumber- answer to daily question, describe frequency of sexual activity or intercourse

g. emotionalCloseness- answer to daily question, describes user satisfaction with the amount of emotional closeness during sexual activity with partner
h. sexualRelationship- answer to daily question, describes user satisfaction with their sexual relationship
i. sexualLife- answer to daily question, describes user's overall satisfaction with their sexual life
j. sexualArousal- answer to daily question, describes how a user would rate their level of sexual arousal
k. sexualArousalConfidence- answer to daily question, describes confidence in sexual arousal
l. lubrication- answer to daily question, decribes how difficult it was to become lubricated
m. lubricationMaintain- answer to daily question, describes difficulty to maintain lubrication until completion of intimacy
n. difficulty - answer to daily question, describes difficulty reaching orgasm
o. satisfaction- answer to daily question, describes satisfaction with ability to orgasm(climax)
p. discomfort- answer to daily question describes comfortability during intercourse

3. Period Table

Period dates for a specific user. Not currently used.
a. pid - period identifier (primary key, auto increment)
b. uid - user identifier (foreign key)
c. mens_start - menstrual cycle (or period) start date
d. mens_end - menstrual cycle (or period) end date

4. Image Table

Image information for uploaded photos. Not currently used.
a. iid - image identifier (primary key, auto increment)
b. uid - user identifier (foreign key)
c. upload_date - date of image upload
d. file_name - file name of image
e. image - image stored as a hex long blob

5. Incident Table

Basic sexual assault incident information. Used by the Sara app. Not used by Luna.
a. incident_id - incident identifier (primary key, auto increment)
b. user_hash - MD5 hashed username
c. incident_loc - location of incident
d. incident_date - date of incident

Luna App Screens

View online at:

http://www.cs.uky.edu/~paulp/luna/screens.html

User Scenarios

Accessing the App and Logging in for a first time user

User goes to the online consent form at the following link: https://luna-app.000webhostapp.com

User reads the consent form and enters in their birthday. If they agree to the study then the user presses the “I consent” button on the website.

If the user is between the ages of 18-25 they will be allowed to download the app and enter into the study. They will then be taken to a page that asks them to provide their UK email address. They will provide this and then press the “register” button. If the user provided a UK email address (ends in @uky.edu or @g.uky.edu), the user continues with step 4. If not a UK email address, an error message is displayed.

The page displays where to download the app. Additionally, an email will be sent to their UK email account. This email will contain a random string that must be used to access the LUNA App once downloaded.

The user will then go to the site where they were told to download the app and download it.

Once the app is downloaded onto their mobile device the user can access the app.

The app will open in a login page and the user will be asked to enter in the random string sent to their UK email account. The user will also be asked to create a username and password that are both 8 characters or longer.

If the random string entered in is correct, and the username and password supplied are valid, then the user will be allowed access to the app and an account will be created for them.

Answering the OnBoarding Questions

After logging in as a first-time user, the user will then be taken to the onboarding questions page within the app. This page contains a series of 9 questions that are essential for the study and for proper app execution.

The user will answer the onboarding questions by sliding from question to question within the onboarding questions page. The questions consist of the following:

Birthday

Cycle length

Period length

Birth control

Reproductive disorder

Last period

Relationship status

Time of the day to input responses to daily questions

Intention of whether to become pregnant.

The user will then press the submit button on the final slide of the onboarding questions page of the app. If all the questions are properly answered then the responses will be sent to the

server and the user's settings will be saved. If a question is left unanswered then an error message alert will be output to the app screen.

The app will then go to the main home page of the app which consists of 3 major tabs.

Daily App Operation

The app can be closed and the mobile device can be powered off but if the app remains on the mobile device and isn't deleted then the user will never be logged out of their account. The user will never again have access to the original login or onboarding questions pages as they are no longer necessary.

When the app is opened it will jump to the home page. This home page will display a calendar and will consist of 3 major app tabs named "Calendar", "Tracker", and "Settings".

To access a specific tab/page within the app a user simply must tap on the tab they want access to.

The major tabs of the app have the following features:

Calendar Tab: The calendar tab will display a calendar of the current month. For a user to change months they simply must slide the calendar left or right. Eventually the calendar will have markers on it indicating when a user's cycle is starting and ending. However, this feature has not yet been added to the app and will be a future enhancement.

Tracker Tab: The tracker tab will allow the user to enter in their responses to the daily questions. Additionally, a future enhancement to the tracker tab will be to allow the user to access their past responses to their daily questions to view how their responses are changing over time.

Settings Tab: The settings tab will allow the user to view and modify their original or latest responses to the onboarding questions. To access this page and change settings a user must first enter in their username and password.

Answering Daily Questions

To respond to the daily questions a user will first click on the "Tracker" tab at the bottom of the main page of the app.

The user will then select the "Daily Questions" option within the "Tracker" page of the app.

The user will then be asked 3 questions:

If they are currently on their period

Their level or desire of sexual interest

If they have engaged in sexual intercourse today

If the user responds "no" to the 3rd question then they can submit their answers to the server by hitting the "submit" button and their responses will be sent if every question is answered properly.

If the user responds "yes" to the 3rd question then they will be given more questions to answer. The user will then be asked 9 more questions:

How often they have engaged in sexual activity or intercourse today

How satisfied they were with the emotional closeness they felt during sexual activity with their partner

How satisfied they are with their sexual relationship with their partner

How satisfied they are with their overall sexual life

Their level of sexual arousal or turn on during sexual activity

How confident they were about becoming sexually aroused during sexual activity

How difficult it was to become lubricated during sexual activity

How difficult it was to maintain lubrication until completion during sexual activity

If they had sexual stimulation or intercourse

If the user responds “no” to the 9th question then they can submit their answers to the server by selecting the “submit” button and their responses will be sent if every question is answered properly.

If the user responds “yes” to the 9th question then they will be given more questions to answer. The user will then be asked 3 more questions:

How difficult was it to reach orgasm

How satisfied they were with their ability to reach orgasm.

If they had intercourse

If the user responds “no” to the 3rd question then they can submit their answers to the server by selecting the “submit” button and their responses will be sent if every question is answered properly.

If the user responds “yes” to the 3rd question then they will be given 1 more question to answer. The user will be asked this final question:

To rate their level or degree of discomfort or pain during intercourse.

The user’s responses will then be sent to the server once the “submit” button is selected if every question is answered properly.

Accessing the Settings Page and Changing Settings

To access the settings page a user must first enter in the username and password they created on the original login page

If the username and password entered in are correct then the app goes to the settings page. If the username or password entered are incorrect then an error message alert is output to the screen.

On the settings page, a user can view their initial/most recent responses to the onboarding questions. The user also has the option to update their responses to the following questions:

Birth Control

Relationship Status

Time of the day to input responses to the daily questions

Intention on whether to become pregnant

Reproductive Disorder

To change a response to an onboarding question a user must toggle the button corresponding to the question they want to update, make the update, and then submit their new answer to the question.

Once one or more question responses have been updated, the user has the option to submit their updated responses to the server. Their new settings are then sent to and saved server side.

Testing

Server Testing from a Browser

Go to:
http://www.cs.uky.edu/~paulp/luna/testserver.html

Luna App Testing

Testing a First-Time user accessing the Luna web site to sign up as a user

New user accesses from a browser:

https://luna-app.000webhostapp.com/

Luna web page	**Luna System Parameter**	**Test Case**	**Pass Condition(s)**	**Fail Condition(s)**
Luna consent form birth date (fail)	enters birth date presses consent button	entered birth date 1 day less than 18 years or 1 day more than 25 years before today's date	Error message shown and user cannot continue “signing” consent form	No error shown and registration process continues -- redirects to register.php
Luna consent form birth date (success)	enters birth date presses consent button	entered birth date 1 day more than 18 years or 1 day less than 25 years before today's date	User redirected to register.php for email address	User cannot continue registration process. Display error message.
New user email address (fail)	enters email address presses submit	email does **not** end in @uky.edu or @g.uky.edu	Error message shown and user cannot continue with the registration process	No error shown and registration process continues-- user row created in User table in database.
New user email address (success)	enters mail address presses submit	email **does** end in @uky.edu or @g.uky.edu	Server sends email with instructions for User to continue registration process-- user row created in User table in database.	User cannot continue registration process. Display error.

Testing Logging into the App as a First-Time user

Luna App Page	Parameters	Test Cases	Pass Condition(s)	Fail Condition(s)
Login Page (Fail)	Username, Password, Random Email Id String	One or more of the 3 parameters are left blank	App outputs a custom alert message saying which field is blank	No custom alert error message is output.
Login Page (Fail)	Username, Password, Random Email Id String	Username and/or Password parameter is less than 8 characters	App outputs a custom alert message saying username and/or password need to be 8 characters or greater.	No custom alert error message is output.
Login Page (Fail)	Username, Password, Random Email Id String	Email ID string input to the app is different from what was received in email. Username and Password pass the 2 tests above.	Server returns an error message to the app. App remains on login page so the user can fix the random email ID input.	No error message returned by the server. App jumps to the onboarding questions page.
Login Page (Success)	Username, Password, Random Email Id String	All 3 tests above are passed.	Server returns that login was successful and UID is returned to the app. UID is stored in local storage of app. Login and password stored in user table for this UID. Hashed version of Username and Password are stored in local storage and sent and stored in the server table. App jumps to the onboarding questions page.	No successful server message or UID received. App does not jump to the onboarding questions page.

Testing Responding to the OnBoarding Questions

Luna App Page	Parameters	Test Cases	Pass Condition(s)	Fail Condition(s)
OnBoarding Questions Page (Fail)	Birthday, Cycle Length, Period Length, Birth Control Type, Last Period, Relationship Status, Time of Day to Respond to Daily Questions, Intention to Become Pregnant, Reproductive Disorder	One or more of the fields to the left is left unanswered	App outputs a custom alert error message for each field left unanswered.	No custom alert error message is output by the app.
OnBoarding Questions Page (Fail)	Birthday, Cycle Length, Period Length, Birth Control Type, Last Period, Relationship Status, Time of Day to Respond to Daily Questions, Intention to Become Pregnant, Reproductive Disorder	Reproductive disorder toggle button is pressed indicating a reproductive disorder but no disorder is input. All other questions are properly answered.	App outputs a custom alert error message stating that a reproductive disorder needs to be input.	No custom alert error message is output by the app.
OnBoarding Questions Page (Success)	Birthday, Cycle Length, Period Length, Birth Control Type, Last Period, Relationship Status, Time of Day to Respond to Daily Questions, Intention to Become Pregnant, Reproductive Disorder	Both tests above are passed.	OnBoarding question answers are sent to and stored in the database user table for this UID. App receives a successful submission message. App jumps to the calendar page.	No successful server message and date isn't stored in the server. App fails to jump to the calendar page.

Testing the Calendar Page

Luna App Page	Parameters	Test Cases	Pass Condition(s)	Fail Condition(s)

Calendar Page (Success)	Calendar	Calendar appears in month format and can be swiped left and right from month to month.	Calendar appears in month format and can be swiped left and right from month to month.	Calendar does not appear in month format and/or cannot be swiped left and right from month to month.

Testing the Tracker Page and Responding to the Daily Questions

Luna App Page	**Parameters**	**Test Cases**	**Pass Condition(s)**	**Fail Condition(s)**
Tracker Page (Fail)	On Period	On Period Question left unanswered.	App outputs a custom alert error message.	App fails to output a custom alert error message.
Tracker Page (Success)	Daily Questions up to the First Toggle Button (Sexual Activity)	First toggle button (Sexual Activity) is left toggled off. The On Period question is answered.	App sends the daily question responses to questions above the first toggle button (Sexual Activity). All other responses are sent as “Undefined”. Server returns a message that the submission was successful.	App fails to send the daily question responses to questions above the first toggle button (Sexual Activity). All other responses are failed to be sent as “Undefined”. Server fails to return a message that the submission was successful.
Tracker Page (Success)	Daily Questions up to the Second Toggle Button (Sexual Stimulation)	First toggle button is toggled on (Sexual Activity). Second toggle button (Sexual Stimulation) remains toggled off. The On Period question is answered.	App sends the daily question responses to questions above the second toggle button (Sexual Stimulation). All other responses are sent as “Undefined”. Server returns a message that the submission was successful.	App fails to send the daily question responses to questions above the second toggle button (Sexual Stimulation). All other responses are failed to be sent as “Undefined”. Server fails to return a message that the submission was successful.
Tracker Page (Success)	Daily Questions up to the Third Toggle Button	First (Sexual Activity) and second toggle buttons (Sexual Stimulation) are toggled on. Third toggle button (Intercourse)	App sends the daily question responses to questions above the third toggle button (Intercourse). All other responses are sent as “Undefined”. Server returns a message that	App fails to send the daily question responses to questions above the third toggle button (Intercourse). All other responses are failed to be sent as “Undefined”. Server fails to return a

		remains toggled off. The On Period question is answered.	the submission was successful.	message that the submission was successful.
Tracker Page (Success)	All Daily Questions (All Toggle Buttons Turned On)	First (Sexual Activity), second (Sexual Stimulation), and third toggle buttons (Intercourse) are toggled on. The On Period question is answered.	App sends the daily question responses to all questions. No responses are sent as “Undefined”. Server returns a message that the submission was successful.	App fails to send the daily question responses to all questions. Server fails to return a message that the submission was successful.
Tracker Page (Success)	All Daily Questions	First (Sexual Activity), second (Sexual Stimulation), and third toggle buttons (Intercourse) are toggled on. The On Period question is answered. Then one of the toggle buttons is switched to off.	App sends the daily question responses to questions above the toggle button switched to off. All other responses are sent as “Undefined”. Server returns a message that the submission was successful.	App fails to send the daily question responses to questions above the toggle button switched to off. All other responses are failed to be sent as “Undefined”. Server fails to return a message that the submission was successful.

*Please note: Responses sent as “Undefined” for the Daily Questions are stored as “0” in the database entry table. *

*Please note: When responses are received by the server, the server creates a new row for that specific UID in the database entry table for this uid with all the responses received. *

Testing the Settings Login Page

Luna App Page	**Parameters**	**Test Cases**	**Pass Condition(s)**	**Fail Condition(s)**
Settings Login Page (Fail)	Username and Password	Username and/or Password field left blank	Custom alert error message output to the app	App fails to output a custom alert error message
Settings Login Page (Fail)	Username and Password	Username and/or Password field incorrect	Custom alert error message output to the app	App fails to output a custom alert error message

Settings Login Page (Success)	Username and Password	Username and Password field correct	App jumps to the actual settings page	App fails to jump to the actual settings page

Testing the Settings Page and Changing Responses to OnBoarding Questions

Luna App Page	**Parameters**	**Test Cases**	**Pass Condition(s)**	**Fail Condition(s)**
Settings Page (Fail)	Birth Control Type, Relationship Status, Time of Day to Respond to Daily Questions, Intention to Become Pregnant, Reproductive Disorder	Toggle certain button(s) to change responses to onboarding questions but leave at least one update response unanswered (The field that will appear beneath the toggle button when the button is turned on).	Custom alert error message output to the app	App fails to output a custom alert error message
Settings Page (Success)	Birth Control Type, Relationship Status, Time of Day to Respond to Daily Questions, Intention to Become Pregnant, Reproductive Disorder	Toggle certain button(s) to change responses to onboarding questions and fill out all update responses for toggled questions (The field that will appear beneath the toggle button when the button is turned on).	OnBoarding question update(s) are sent to and stored in the database user table for this UID. App receives a successful submission message. App jumps back to the settings login page.	OnBoarding question update(s) are failed to be sent to and stored in the server. App fails to receive a successful submission message. App fails to jump back to the settings login page.

Testing Notes

When the app is running (simulator or device), operation messages appear in the Xcode debug window (lower right). The console.log messages appear in this window. You can add messages to appear in this window by adding console.log statements in the typescript code.
Whenever a change is made to any app code, run the three steps listed in the installation for building and testing the app..
We have found that the simulators built-in to Xcode test all functions of Luna the same as the physical devices tested. One exception is that the simulator retains the

simulated app's local storage. As explained in the design, the user ID (uid) is retained in the app local storage so that the user can use the app to report daily questions without providing a logon user name and password. When using the simulator, this means that after first use, the simulator goes to the calendar page, skipping the logon screen.
To clear the simulator local storage, execute this command from the mac terminal (the simulator must be closed):

xcrun simctl erase all

Future Enhancements

Apple Developer Program

First and foremost you must be a member of the Apple Developer Program.

To enroll in the Apple Developer Enterprise Program, your Apple ID must be associated with an email address that uses your organization's domain name.

Go to https://developer.apple.com/programs/

Click ‘Enroll’ in top right

Scroll to bottom of page, click ‘Start Your Enrollment’

Sign in with your Apple ID)

Select Entity Type ‘Company/Organization’ click continue

Please note: In order for your company name to be listed as the seller on the App Store, your company must be recognized as a legal entity in your country. Sole proprietors and single person companies located in countries where they are not recognized as legal entities will be enrolled and be listed for download under the enrollee’s personal legal name, if selected for distribution by Apple.

continue following account setup instructions

once you’re enrollment has been verified go to https://developer.apple.com/support/purchase-activation/ for pricing and purchases support.

Please note: The Apple Developer Program annual fee is 99 USD and the Apple Developer Enterprise Program annual fee is 299 USD, in local currency where available. Prices may vary by region and are listed in local currency during the enrollment process.

Developers need to have an Apple ID (free) to get Xcode for Luna app testing. However, to make the Luna iOS app available in the Apple store, the customer must enroll in the Apple Developer Enterprise Program ($99/year).

Android Version

The Ionic development framework can be used to develop iOS and Android apps. Due to time constraints, and lack of Android devices for testing, we have only used it for iOS development and testing. According to the Ionic documentation, it is a simple matter to create an Android app, using the same code used for iOS development.

Remind User to Answer Daily Questions (and only once per day)

A reminder sent to the user by the app to answer the daily questions if they have not yet been answered by the time the user specified to answer them in the day.

Only allow the User to Respond to the Daily Questions Once per Day: Currently a user can respond to the

daily questions multiple times a day and all responses will be stored in the server. In the future, a user needs to be allowed to respond to the daily questions once per day

Research Queries to the Data Base

For the customer to use the information that users will supply, and that will be put in the Luna Data base, queries will have to be supplied that a person not experienced in data base operation and SQL queries can use.

App Enhancements

Calendar Enhancements: Add in the event markers to the calendar to signal the beginning and end of a user's cycle. Will then need to add a legend to the calendar to show which symbols/colors represent what on the calendar. Additionally, the calendar page could also give some type of countdown that lists the number of days until the next cycle.

Settings page enhancements:

Update the Last Period in the Settings Page: In the settings page, the last period date is an output. Once the app is operational this needs to be determined by a user's response to the "Are you on your period" daily question. Currently the app will output the original response to this onboarding questions forever and there is no way for the user to change their response manually since this needs to be determined by the app itself.

Update the Cycle Length in the Settings Page: In the settings page, the cycle length is an output. Once the app is operational this needs to be determined by a user's responses to the "Are you on your period" daily question. Currently the app will output the original response to this onboarding questions forever and there is no way for the user to change their response manually since this needs to be determined by the app itself. Additionally, the app could also compare the user's original cycle length (what they originally entered when inputting their responses to the daily questions) to the cycle length the app has determined a user has.

Update the Period Length in the Settings Page: In the settings page, the period length is an output. Once the app is operational this needs to be determined by a user's responses to the "Are you on your period" daily question. Currently the app will output the original response to this onboarding questions forever and there is no way for the user to change their response manually since this needs to be determined by the app itself. Additionally, the app could also compare the user's original period length (what they originally entered when inputting their responses to the daily questions) to the period length the app has determined a user has.

Progress Tracker for the Tracker Page: Make a way for the user to view their past responses to the daily questions. This would allow the user to see how their responses are changing over time. Note that this feature would probably require a login page like what is currently used for the settings page due to security reasons.

Link the Luna App and Sara App: Allow Luna and Sara to communicate with one another. For example, if a user submits a sexual assault through Sara then they should get an update in Luna saying they submitted a sexual assault. This feature would need to be password protected only allowing the user themselves to see the report alert. The user would therefore need to be asked to sign in again.

Error Log and Support Notification

Log errors that can't be fixed by a user in a server error log table so that a support person can work to resolve these issues. Send email to a customer account to notify the support person.

Password/username changes

Password/Username Changes: Currently if a user forgets their password or username then they can delete the app off their phone and re-download the app. If they remember their random email ID then they could just re-log into the app and their old data would still be stored in the server. However, if the user does not remember their random email ID string then they would lose their account when downloading a new app. In the future, there needs to be some link for a user to select on the settings page if they forget either their username or password with a way to update them. However, since the user's email account is not stored within the database we would probably have to ask the user to input their email again so they can be sent a new form allowing them to update their username and password.

Conclusions

This past semester the amount of things we learned really is innumerable. Here is a brief list of just some of the things we learned:

Ionic Framework: We learned how to work with the Ionic Framework, particularly Ionic version 2. Ionic was at the core of app creation with its built-in dependencies and commands that made it simple to add features into our app.

Xcode: To do the final build of the app and transport it to a mobile device like an iPhone or an iPad Xcode was used.

NodeJS: NodeJS was used within the app for server communication and the HTTP protocols. By incorporating this feature into our app, we allowed it to communicate with our server and database.

Databases: To have a working project one of our major project constraints was to have a working database that could receive and store the user info sent from the app. Throughout the semester we built a database using webhost that works in sync with our Luna app. Our customer will have access to this database which is needed for research purposes.

App Making: We learned about app making in general. We learned that Ionic can be used to make apps for iOS, Android, and Windows operating systems. We learned that apps are typically made with a combination of Typescript, HTML, and Sass code for Ionic (in Xcode an app can be made directly for an iOS device typically using just use the Swift language)

Security: Since we were sending sensitive information back and forth between the app and the server and storing it we learned about the Md5 protocol and hashing (one-way encryption). Security was achieved by storing both the username and the password in a hashed version server side and sending a random user ID to the app. This user ID was then stored it in local storage to identify app users when sending data to the server.

Local Storage: We learned about app local storage and how it could be used to hold app info within the user's mobile device. Using local storage allowed us to bypass the login page except for initial login, making usability much simpler. We also learned that by using local storage we could use fewer server calls to the database since some info could be stored directly on the device in the app making the overall project design much simpler.

Network Communication: We learned about the HTTP protocol and how data can be sent over a network to a server. This was an essential feature of our app since we wanted user info to be stored in the server so the customer could access essential research data.

Product Development: This class was set up to similar to a how a product design team would operate at a major corporation. We learned much about product development as we had weekly meetings with our customer and also met twice weekly (sometimes more) as a group to discuss product details, implementations, and potential challenges.

Testing: We learned the importance of creating an efficient and well documented testing plan that is easy to follow. By creating a complete testing plan, it not only made it easy to test the app but will also make it easier for future groups that may access or modify our product.

Documentation: We learned the importance of proper documentation and how it is necessary for any major project like the one we undertook. Over the last two weeks of the semester we have primarily been concerned with documenting everything in our app. We have documented our

source code, testing procedure, user scenarios, installation, etc. This will help any future developers who may view our designs and make modifications to the app and database.

Teamwork: We learned the importance of teamwork and communication when working on any major project with more than one team member. To stay up to date on our latest project developments it was essential to use email to communicate between team members and let them know what was going on.

As you can see, throughout the semester we learned how to work with a wide variety of technologies and how to incorporate them all into one final working project. It seemed like every week we were faced with a new challenge that could potentially derail our entire project. However, through perseverance our group always managed to find a solution, implement it into the design, and solve the problem before tackling the next obstacle.

Although we couldn’t meet all customer requirements for the app, we have created a very solid groundwork that a future team will be able to use to finish the Luna project. We have virtually all essential features of the app incorporated except for the Calendar component, and the database is functioning properly to record user data. Our design and documentation make it easy to determine how the Luna product functions.

Installation

The Summer 2017 delivery (root directory LunaShip) has the components (subdirectories):

Documentation
LunaApp
ServerDatabase

The Documentation directory contains the product documentation:

Luna report (this document) in HTML, and docx format
Calendar documentation (pdf)

The LunaApp directory in the LunaShip delivery zip file is the home directory for the Luna app, that contains all the needed files for the app creation and maintenance. See the instructions for the app installation.
The ServerDatabase directory contains the server files, and database creation file. See those instructions.

Luna App Installation of the Summer 2017 Luna Product

When you unzip the product, the LunaApp directory (under the home LunaShip directory) is the home directory for Ionic commands to build and maintain the app. All the commands below are executed in a Mac terminal from that directory that will be called the home directory below.
The app is built using the Ionic framework and tools. It is tested (in a simulator on the Mac, or in an iPhone or iPad) using the Xcode IDE. During this term, the Sublime IDE was used to edit source files on the Mac. It was used because it recognizes TypeScript files (.ts) and formats them for the developer.
Product release levels used during the Summer 2017 term:

Mac OS Sierra 10.12.5
Ionic framework version 2
Xcode 8.3 (free)
Sublime (unregistered free version, Stable Channel Build 3126)
iPad 10.3.2
iPhone 6 10.3.2

The npm project manager tool is used to install packages needed by ionic.
Xcode must be used to test the app, either using one of its device simulators, or the actual device attached to the Mac. Any editor can be used to maintain the source code.
Instructions to install the Luna ionic build from Summer 2017
This installation installs the ionic product (complete directory) created during Summer 2017. A new ionic product can be created in a new directory. This takes more steps, and need not be done by the developers who will be supporting the Summer 2017 Luna product.

- The instructions used by the Summer 2017 developers to create a new Luna ionic product are below
- Instructions for the Web Host server/database for Luna are below

Ionic can be used to build an iOS or Android app. During summer 2017, we only built and tested the iOS Luna app. However app code development can be done on

Windows, and was done so during summer 2017.
Instructions for developing the Luna app on Windows are below

Adding a new page to the Luna app instructions are below

Testing Notes are below

Luna Installation on a Mac

These directions install the app, created during the summer term 2017. A new Luna ionic project can be created. You may want to do this if there is a major ionic upgrade, or to change the name of the product (Luna) that appears below the icon on the device. See those instructions below..

Execute from a Mac terminal:

sudo npm install -g cordova ionic

This installs ionic. "sudo" runs the command as the system administrator.

Copy the Luna_app directory from the LunaShip directory to where you want to install the Luna appproject. The Luna_app directory is the home directory to execute the commands to build and execute the
Luna app using Xcode to test the app on its simulator, or an iPhone, or iPad device connected to your mac.

Run the following commands from a Mac terminal in the ionic project home directory

(the Luna directoryyou created in step 2): **sudo ionic cordova plugin add**

cordova-plugin-http You get the message:

The plugin @ionic/cli-plugin-cordova is not installed. Would you like to install it and continue? (Y/n) answer
Y

Add app local storage support by running the following commands:

sudo npm install @ionic/storage@2.0.0 --save --save-exact

sudo npm uninstall --save @ionic/storage

sudo npm install --save @ionic/storage

Install momentJS for the Calendar by running the following command:

sudo npm install --save moment

Install the md5 hashing protocol by running the following command:

sudo npm install ts-md5

Create the ios support:

sudo cordova platform add ios

Install cordova HTTP support:

sudo ionic cordova plugin add cordova-plugin-http

Building and Executing Luna

Using the Xcode simulator as the device:

The three commands to execute from the terminal in the Luna home directory (luna_app), that build, then executes Xcode to test:

ionic serve

brings up a browser window with the Luna login screen. If syntax errors, they are displayed.
ctl-c in the terminal to enter next command.

cordova build ios DO NOT HAVE A DEVICE PLUGGED INTO THE MAC

open platforms/ios/Luna.xcodeproj

Commmand 3 starts Xcode with the Luna Xcode project. Choose the simulator to run (upper left by Luna name, pull down list). Then press run icon (to its left, left "arrow")
"Build Succeeded" window appears, simulator starts, displays Luna login page.
These three commands are run whenever you modify the app code.

Using Xcode to install the app on a device (iPhone, iPad)

Plug the device into the Mac. Its name should appear in Xcode device window (click pulldown to right ofLuna icon). Click it.
Press Xcode start build icon.You should see "build succeeded" window, and request to update keychain.
On the device, from the settings app, press device management, and allow this app developer access.
The Luna icon should appear on a screen.
Note:

We got "signing" errors, and "provisioning" errors when we first tested devices from Xcode. To eliminate errors like "Failed to create provisioning profile", or signing errors, we had to do on Xcode:
Show the project navigator

on the left pane: view--

>navigators-->show

project navigator Click on

Luna (top in navigator

pane)

in identity: replace bundle name with something else (the string "paulp" works)
in signing: click automatically manage signing
Team: should display your Apple developer ID
Your Apple developer ID can be displayed: Xcode-->Preferences-->accounts
Apple developer IDs are free, and you must have one to develop an app for Apple devices.
See the instructions on becoming an Apple developer under Future Enhancements.

Server and Database

The server and database are hosted on the WebHost free hosting site. The directory structure for server files:

includes - directory for non-publicly accessible helper files

constants.php - database connection strings (other constants may go here)

dbconnection.php - MySQL database connection functions

dboperation.php - MySQL database functions for server programs

funcs.php - commonly used functions

public_html - directory for:

htaccess - server access rules

index.php - consent form (main page accessed by index.html)

LunaAppConsentForm.pdf - user downloadable pdf of user consent form

LunaImage.png - Luna image for main app page

php_errors.log - used by webhost to log errors

public_html/api/v1 - server programs accessible by the app and index.html

auth.php - app interface for login for new user

register.php - used in new user process, creates entry in user table

changeOnboard.php - changes onboard question answers
addDaily.php - adds new row to entry table for daily question answers
changePassword.php - changes user's password (not used)

deleteUser.php - deletes user (not used)

logIncident.php - logs sexual assault incident (used by SARA)

See the design for how these programs are used, and for the description of the database tables.
NOTE: If changes are made to the index.php file for the consent form, be sure to create a new
LunaAppConsentForm.pdf file and upload it to the public_html directory in Webhost.

Enter the following URL in a browser: https://www.000webhost.com/cpanel-login and enter the following to access the server PHP files and database:

Username: joshua.cockerill1@gmail.com
Password:
This is the username (email address of spring semester student) used for the Spring/Summer 2017 classes, to be changed to a customer email address. Contact the customer for the password.

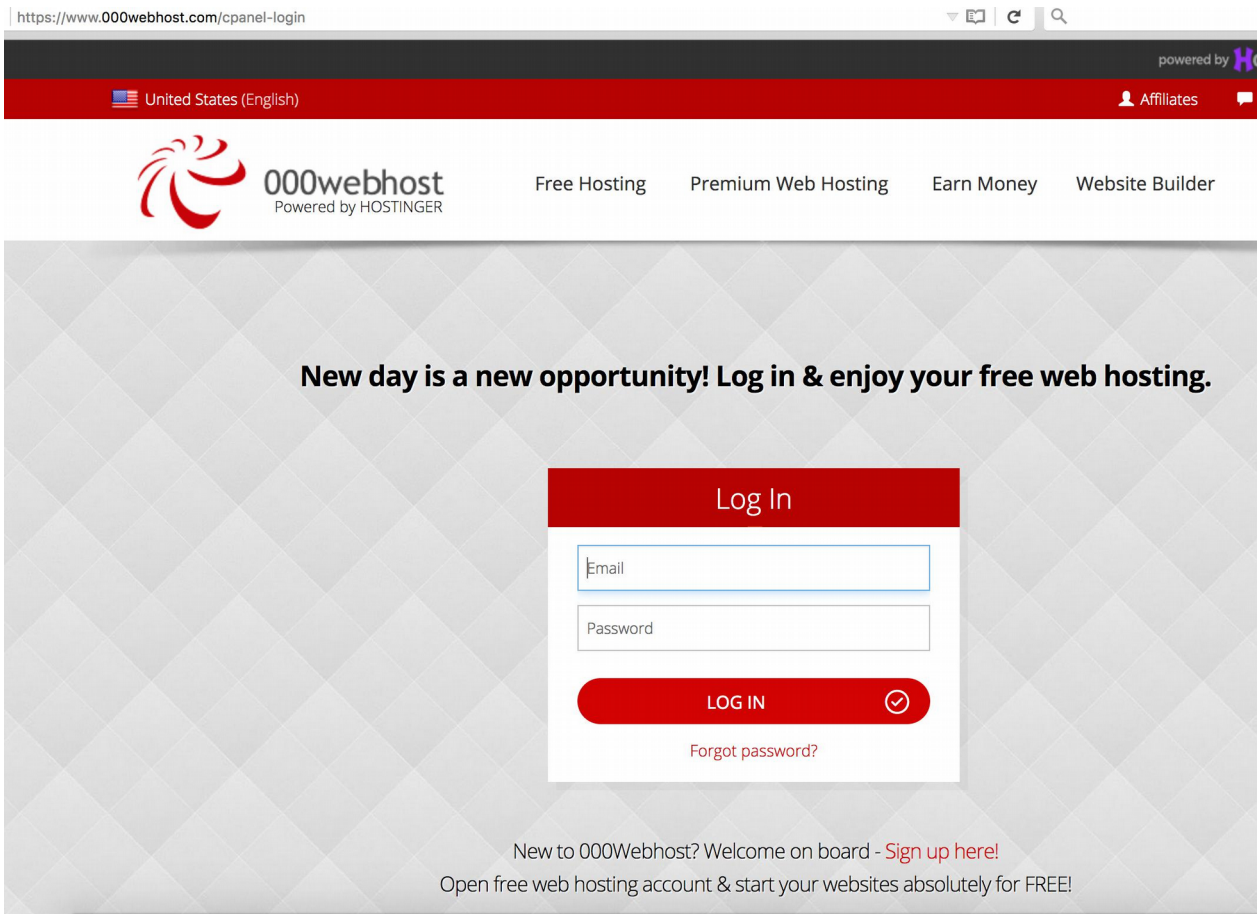


Figure 4.1 – Web API Host Login

Upon logging in, the main Webhost page is displayed. By clicking “Manage database”, located in the upper navigation bar, the administrator will reach a page where they may choose to manage the database for the Luna app.

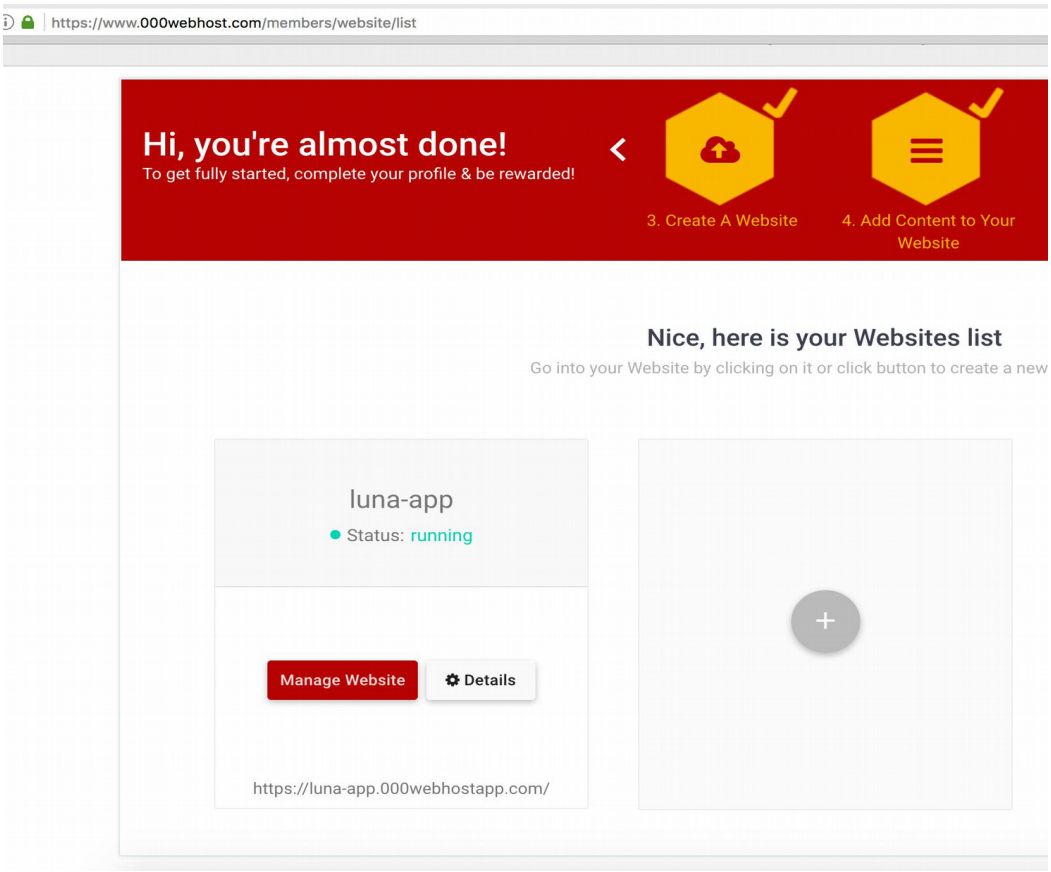


Figure 4.2 – Web API Host Main Page

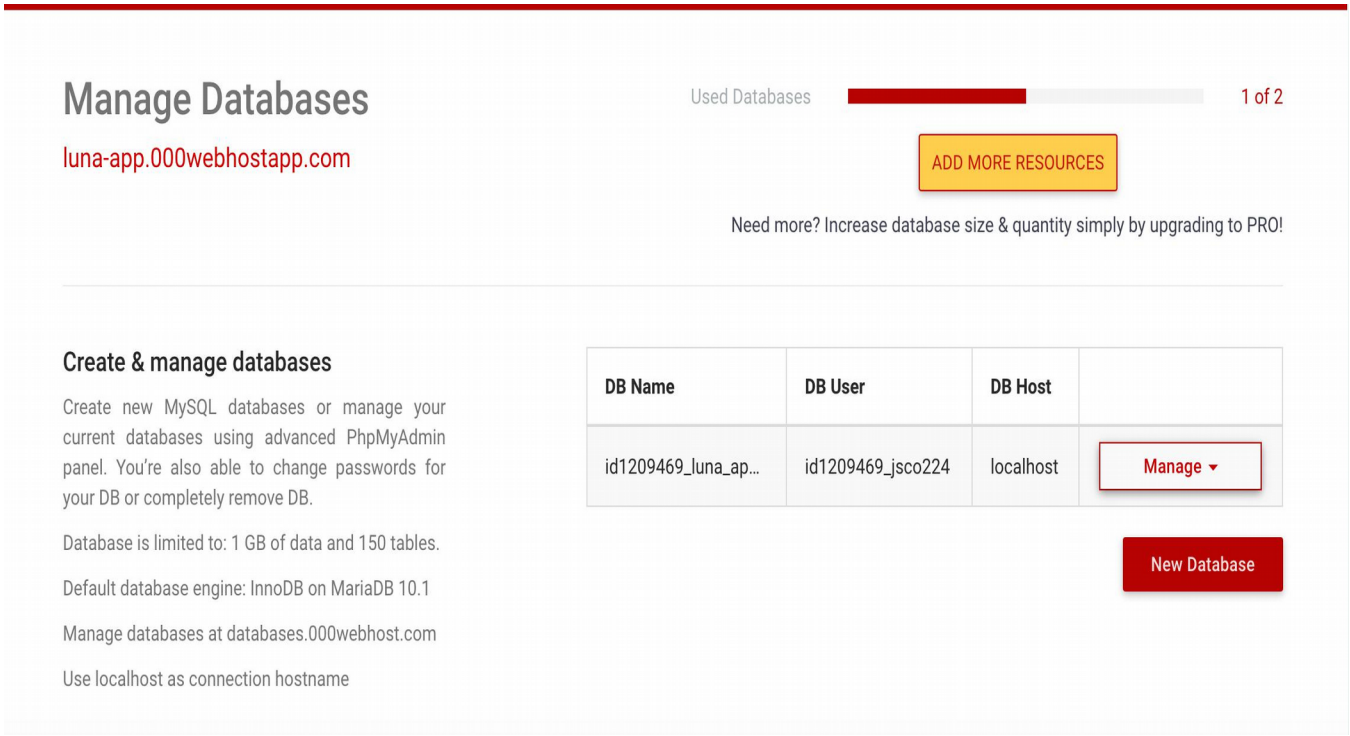


Figure 4.3 – Web API Host Database Access

To view and maintain the database, click the “Manage” dropdown and select “PhpMyAdmin”. You can view and update all the Luna tables.
The database is accessed by selecting “manage” and selecting “phpMyAdmin” from

the dropdown.

The username to the phpMyAdmin site:

Username: id1209469_jsco224
Password:
Contact the customer for the password. The username is created by WebHost, and need not be changed.

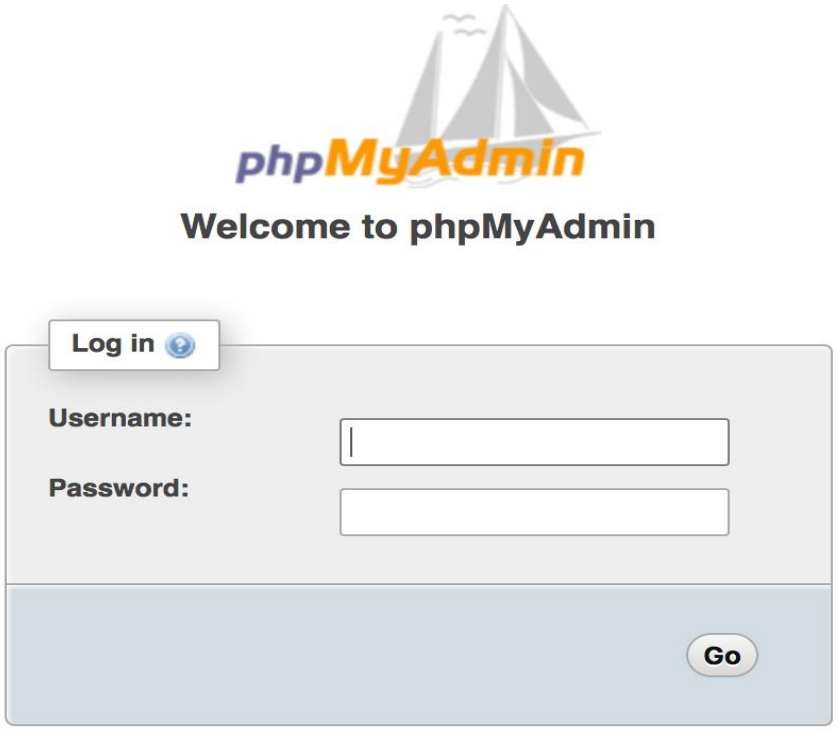


Figure 4.4 – PhpMyAdmin Login Page

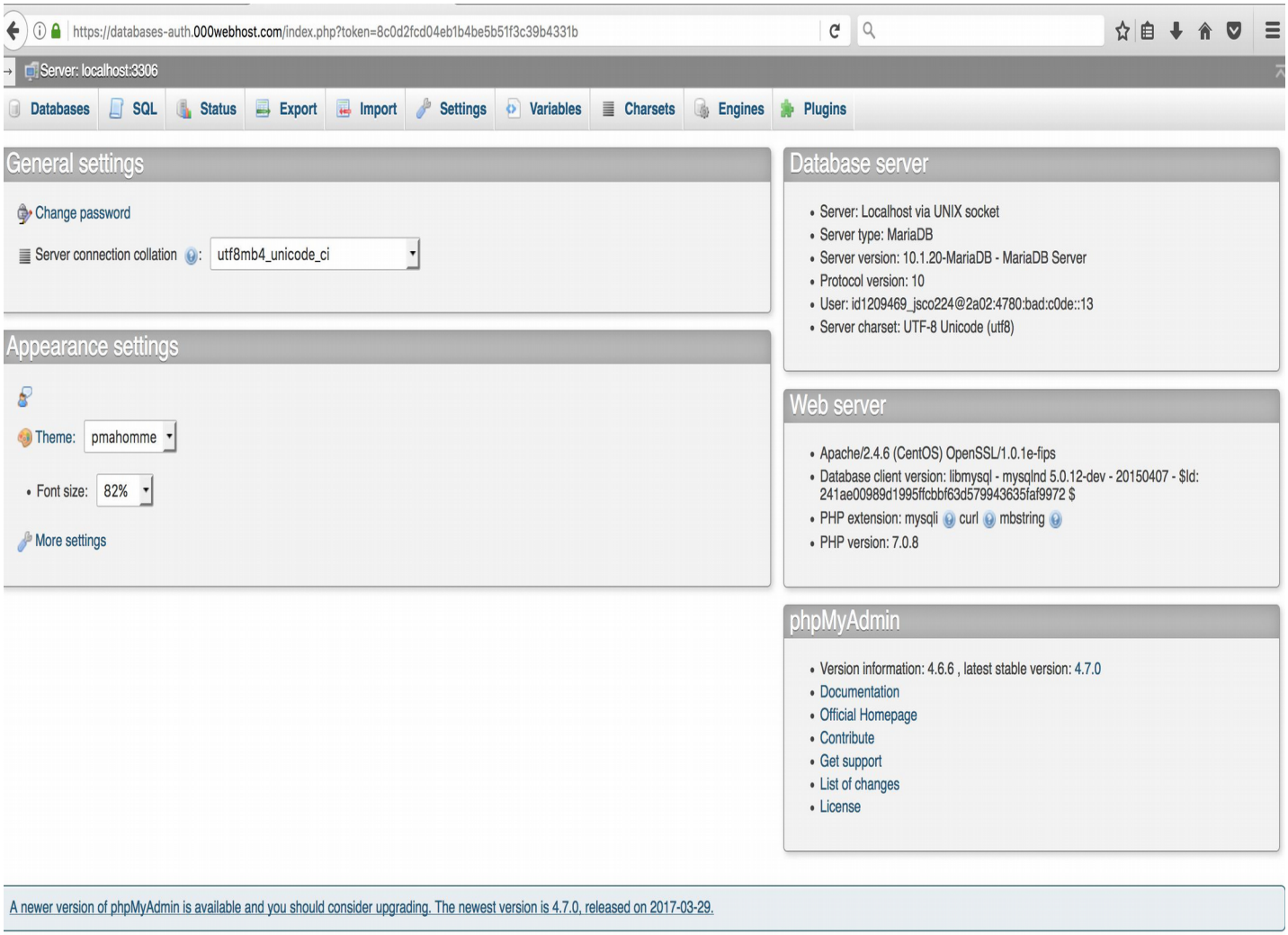


Figure 4.5 – PhpMyAdmin Main Page

When changed, the Luna server files are uploaded to the 000webhost FTP server. Upon logging into the API, you will see the main page. By clicking “File manager” and then the “Upload files now” button, you will reach a page where you can choose to navigate, modify access, and update the Luna server files.

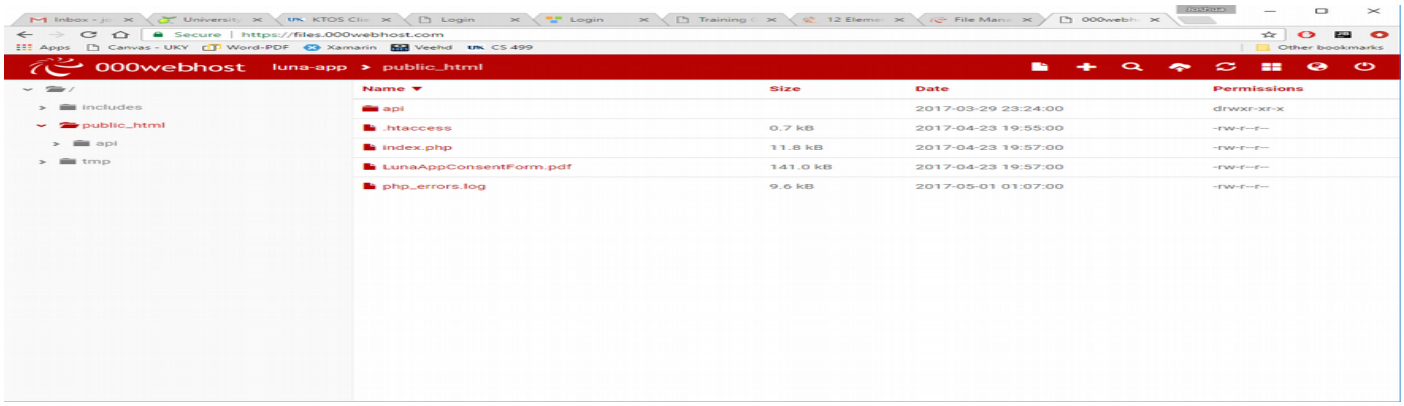


Figure 4.6 – FTP Page

Luna Installation on a Mac (Creating a new ionic project)

You can install the summer term Luna app, using the Luna ionic project created during that term, following the previous directions. These directions will create a new ionic project in a new directory. You may want to do this if there is an ionic upgrade, or you want to change the app name.

From the directory where you will create the Luna home directory, execute from a Mac terminal:

npm install -g cordova ionic (Installs ionic)

From this directory execute:

ionic start Luna tabs

Do not make directory Luna. This

command makes the directory Luna

You will be asked:

"Link this app to your Ionic Dashboard to use tools like Ionic View?"
Ionic sells development tools (Ionic View) that are not needed to build and maintain Luna. Answer no.
The app name this will be displayed on the iPhone or iPad under the icon is taken from line 3 of the file Config.xml in the created Luna directory:
<name>MyApp</name>
The default is MyApp. **Change it to Luna NOW**. Once the following steps are taken, ionic won't let you change it without creating errors.

Copy source files files for Luna

pages directory: Delete the created pages directory for the sample app, replace

with the Luna pages directory from the directory: LunaShip/Luna_app/src/pages

app directory: Ionic has two files that define the pages:
Luna/src/app/app.module.ts
Luna/src/app.component.ts
Replace these with the ones in LunaShip/Luna_app/src/app

Replace the **theme file** in the theme directory: Luna/src/themevariables.scss

with the one from LunaShip/Luna_app/src/theme

Run the following commands from a Mac terminal in the ionic project home directory (the Luna directory you created in step 2):

ionic cordova plugin

add cordova-plugin-

http You get the

message:

The plugin @ionic/cli-plugin-cordova is not installed. Would you like to install it and

continue? (Y/n) answer Y

Add local storage by running the following commands:**npm install**

@ionic/storage@2.0.0 --save --save-exact

npm uninstall --save @ionic/storage

npm install --save @ionic/storage

Install momentJS for the Calendar by running the following command:

npm install --save moment

Install the md5 hashing protocol by running the following command:

npm install ts-md5

Create the ios support:

cordova platform add ios

Replace **app icons**, copy from:

LunaShip/Luna/resources/ios/icon directory

to the created Luna icon directory

Note: Creating the ionic app icons is explained here.

Install cordova HTTP support:

ionic cordova plugin add cordova-plugin-http

Copy **node_modules** directories from: LunaShip/Luna/node_modules/@ionic-native/:

http

keyboard

sql-lite

Building and Executing Luna

On the Xcode simulator:

The three commands to execute from the terminal in the Luna home directory, that build, then executes Xcode to test:

ionic serve

brings up a browser window with the Luna login screen. If syntax errors, they are displayed.

ctl-c in the terminal to enter next command.

cordova build ios DO NOT HAVE DEVICE PLUGGED INTO MAC

open platforms/ios/Luna.xcodeproj

Commmand 3 starts Xcode with the Luna Xcode project. Choose the simulator to run (upper left by Luna name, pull down list). Then press

run icon (to its left, left "arrow")

"Build Succeeded" window appears, simulator starts, displays Luna login page.

These three commands are run whenever you modify the app code.

On a device (iPhone, iPad)

Plug the device into the Mac. Its name should appear in Xcode device window (click pulldown to right of Luna icon). Click it.

Press Xcode start build icon.You should see "build succeeded" window, and request to update key chain.

On the device, from the settings app, press device management, and allow this app developer access. The Luna icon should appear ona screen.

Note:

We got "signing" errors, and "provisioning" errors when we first tested devices from Xcode. To eliminate errors like "Failed to create provisioning profile", or signing errors, we had to:
Show the project

navigator on the

left pane: view--

>navigators--

>show project

navigator Click on

Luna (top in

navigator pane)

in identity: replace bundle name with something else (qwert)
in signing: click automatically manage signing
Team: should display your Apple developer ID
Your Apple developer ID can be displayed: Xcode-->Preferences-->accounts

Installing Luna on Windows OS

Step 1: Downloaded the Microsoft Visual Studios Community. After it has downloaded and installed you will then see multiple programs within Visual Studios you may install. Select and install both the basic visual studios editor (note: this will be installed automatically!) as well as the "Mobile App development environment with Javascript". Install this by checking its box and then selecting the install button at the bottom of the page. Note that the Visual Studios Editor is free to UK CS students and can be installed at the following link: https://www.cs.uky.edu/csportal. Once logged on, click the MSDNAA link on the left.

Step 2: Download and install NodeJS. NodeJS can be installed at the following link https://nodejs.org/en/. Choose version v6.11.0 LTS. This is the version that is recommended for most users.

Step 3: Download and install "GitBash" or a similiar termial. Having a terminal window similiar to a Linux environment is crucial and GitBash appears to be recommended the most for Windows users as the Windows OS doesn't readily/easily have a terminal available as a MAC OS does. GitBash can be downloaded and installed at the following link: https://git-scm.com/downloads. After going to this site

click on the Windows link and the installation process should begin.

Step 4: Download and install Ionic by opening up a GitBash terminal window and running the command "npm install -g ionic cordova". Note that if you are asked to update anything while in the terminal while the installation process is going select "yes". This should install Ionic, the Ionic 2 framework, and Cordova which will allow for the creation of applications. Run from the terminal window: "npm rebuild node-sass". This command is needed when using an app built on Mac, on a Windows OS, to have the correct version of nodeJS running.

Step 5: Install the Ionic templates to Visual Studios. Note that the templates you have to install with Visual Studios 2017 is the Ionic 2 Framework even though it is still in its Beta development stages. In order to properly install the Ionic templates to Visual Studios follow the process shown at the following link: https://taco.visualstudio.com/en-us/docs/tutorial-ionic/

Step 6: Copy over the LunaShip.zip file that contains the directory Luna_app (the app code). Then use the GitBash terminal and the unzip command to unzip all the files. Note that you can't simply extract all the files as this won't work creating paths that are too long. Unzip the file by using the 'cd' Linux command to change directories into where the zip file is stored. Then use the command "unzip LunaShip.zip" to unzip the file.

Step 7. Open a GitBash terminal and use the 'cd' or change directory command to step into your Luna_app folder that will be created after unzipping LunaShip.zip. This folder contains all the necessary files for the LUNA app to run. Then execute the "ionic serve" command once in the correct directory to open up the LUNA app and get it running. Note that while doing this you may recieve some notifications to update/upgrade some of your environment features. Say yes to all updates. The app will then open up correctly in a browser.

Note: There is a way to open apps like Luna directly from Visual Studios but so far I have been unable to do this as it involves copying over 50 folders into an Ionic Visual Studios Project in order to get all the dependencies correct. I am still working on a way to do this easily.

However, by using the "ionic serve" command the Ionic app opens up in a browser. You can then open up that app's specific files/pages in an editor like Visual Studios and make changes to them. After saving these files you will see the app update itself in real time without having to reload it each time a change is made!

Note that in order to open the app up in lab mode and see how it will look on different devices (iOS, Android, and Windows) instead of executing the "ionic serve" command execute the "ionic serve -l" command.

Steps to Adding a New Page to an Ionic Project

Step 1. Within the src/pages folder of the project add a new folder named “newPageName” that will contain the 3 files needed to properly create a new page. This folder will contain an HTML file (.html), a Typescript file (.ts), and a Stylesheet

file (.scss).

Step 2. Create the 3 files listed above within the “newPageName” folder. Name the 3 new files as follows:

newPageName.html
newPageName.ts
newPageName.scss

Step 3. Within the “newPageName.ts” file you need to have the following:

```
@Component({
  selector: 'page-newPageName',
  templateUrl: 'newPageName.html'
        })
export class NewPageNamePage { … where the file’s code is written … }
```

Step 4. If you are using the “Tabs” component/feature within your app read this step. If you are not then you may skip over this step.

Within the tabs.ts file you need the following:

```
import { NewPageNamePage } from '../newPageName/newPageName';
```

Note: this is done at the top of the tabs.ts file

If you are wanting to make this new page a root tab of the app:

```
Tab#Root: any = NewPageNamePage;
```

i. Note: # will be replaced by the specific # of the tab you want this page to be.

Step 5. Within the app.module.ts file you need to do the following:

```
import { NewPageNamePage } from '../pages/newPageName/newPageName';
```

a. Note: this is done at the top of the app.module.ts file

```
@NgModule({
  declarations: [
  MyApp,
  Page Name 1,
  Page Name 2,
  NewPageNamePage,
  TabsPage
      ],
```

Note: “Page Name 1” and “Page Name 2” will be replaced with actual page names. There also could be many more pages within your app…

Note: If the Tabs feature/component is not used in your app then don’t include the “TabsPage”. 3. entryComponents: [

```
  MyApp,
  Page Name 1,
  Page Name 2,
  NewPageNamePage,
  TabsPage
      ],
```

a. Note: “Page Name 1” and “Page Name 2” will be replaced with actual page names. There also could be many more pages within your app…
b. Note: If the Tabs feature/component is not used in your app then don’t include the “TabsPage”.

Step 6. This step is only used if you want to make the new page you are creating the root page of the app. In other words, this is the original page the app will open up in. If not, then skip this step.

Within the app.component.ts you will need the following:

1. import { NewPageNamePage } from '../pages/newPageName/newPageName';
    a. Note: this is done at the top of the app.component.ts file
2. rootPage = NewPageNamePage;
    a. Note: this is done within the export class. So:
    b. export class MyApp { …
        rootPage = NewPageNamePage; … }

Testing Notes

When the app is running (simulator or device), operation messages appear in the Xcode debug window (lower right). The console.log messages appear in this window. You can add messages to appear in this window by adding console.log statements in the typescript code.

Whenever a change is made to any app code, run the three steps listed in the installation for building and testing the app..

We have found that the simulators built-in to Xcode test all functions of Luna the same as the physical devices tested. One exception is that the simulator retains the simulated app's local storage. As explained in the design, the user ID (uid) is retained in the app local storage so that the user can use the app to report daily questions without providing a logon user name and password. When using the simulator, this means that after first use, the simulator goes to the calendar page, skipping the logon screen.

To clear the simulator local storage, execute this command from the mac terminal (the simulator must be closed):

xcrun simctl erase all

References

Xcode (we used release 8)
Sublime text editor
Steps to test a device from Xcode
Apple developer information
Webhost (Luna server and database hosting site)
Ionic development references we used

Helpful Commands & Resources for Ionic

Ionic framework:*https://ionicframework.com/docs/*
Resource for Ionic Local Storage: https://ionicframework.com/docs/storage/
Date-Time Picker Implementation:
https://ionicframework.com/docs//api/components/datetime/DateTime/
Hiding elements of the HTML page based on variables:
https://scotch.io/tutorials/how-to-use-ngshow-and-nghide
Page Navigation: https://www.joshmorony.com/a-simple-guide-to-navigation-in-ionic-2/
Other calendar options then the one we used can be found at: http://angular-ui.github.io/ui-calendar/

https://thielcole.github.io/ionic2/2017/02/03/Ionic2-Calendar.html

A good video that helps to explain local storage can be found at:
https://www.joshmorony.com/a-simple-guide-to-saving-data-in-ionic-2/
Creating an Ionic Picker: https://www.npmjs.com/package/ion-multi-picker

https://www.npmjs.com/package/cordova-wheel-selector-plugin#android

https://github.com/mmnaseri/proton.multi-list-picker

Ionic Market-Place Picker's: https://ionicframework.com/docs/native/date-picker/

https://ionicframework.com/docs/native/wheelselector-plugin/

https://ionicframework.com/docs//components/#datetime

Command to serve/run the app in a browser: ionic serve
Command to serve/run the app in a browser for multiple environments (ios, android, and windows): ionic serve -l

Command to create the actual Ionic app: ionic cordova build platform_name

Platform_name could be: ios or android
Command to add a platform to Ionic: ionic cordova platform add platform_name

Platform_name could be: ios or android
Command to check requirements for what's needed for the build to take place: cordova requirements

When changing a file name and dependencies you must change the following 5 things:

Name of the folder with 3 files in it

Name of the 3 files within the folder.

Names within the file_name.ts file

Names within the tabs.ts file

Names within the app.module.ts file
Note: Plug-ins cannot be tested in the browser, they only work with an actual build test.
Note: If Cordova, npm or any dependencies have an update and the app no longer works you may have to uninstall Cordova using the following command: npm uninstall -g cordova

Then re-install Cordova using the command: npm install -g ionic cordova

Then say yes to all Cordova updates and the app should work again!!!
Note: The calendar.service.ts is most likely the key to adding eventMarkers and events to the calendar.
Note: The displayUID 3 files are not actually used in the app but are for debugging purposes to display a user's UID.
Note: The app goes to the settingsLogin page before going to the actual settings page.