Project Management Report

February 21, 2018

Team 3 (Jack Taylor, Mitchel Smith, Elan Kainen, Jason Wen)

Introduction:

This report describes Team 3's management plan for its project to develop an iOS application to connect travel buddies based on interests and trip scheduling.

1. Expected Level of Effort in Person-Months

Software Component	Estimated LOC
User Registration/Sign In/ Sign Out	750
User Profile Page	200
Trip Creation Page	400
Trip Scheduling/Matching	600
Built-In Server	700
Integration w/ Google Maps API	450
Social Network Functionality (Friends/Chatroom/etc.)	600
SUM	3700

Effort Calculation (Using the COCOMO II Model)

$E = a*LOC^b$ $a = 3.9$ $b = 0.91$ $LOC = 3.7$ kLOC

$E = 12.83$ Person-Months

We believe that this estimate is low because it is measuring in person months rather than student months. To account for this, we organized our goals in a way that has more complex, optional features being completed in later iterations of the project. It is also important to note that our LOC estimations are quite rough.

2. Overall High-Level Schedule for the Project

Task	Duration (days)	Work (person-days)	Start date	End date
Iteration 1: Built-in Server, User registration/sign in/sign out	36 (including Spring Break)	150	Feb. 22	March 30

Iteration 2: User Profile Page, Trip Creation	11	62	March 31	April 10
Iteration 3: Google Maps API, trip scheduling/matching	18	109	April 11	April 29
Iteration 4: Social network functionality	10	62	April 30	May 9

Note:

- Iteration 4 will include extra embellishments if and only if the previous 3 iterations are sufficiently completed on schedule. This high-level schedule for the project is subject to change if each goal is not met in the allotted time.

3. Quality Plan

We plan on running quality control tasks throughout each iteration in order to ensure that each component of the software project is being appropriately built-out. Obviously our quality goal for each iteration, and the project as a whole, is to have as few defects as possible in the code to ensure optimum functionality. However, setting an acceptance criteria quantitatively for each component is a rather fruitless task because, at this point, we have no reference point to state our task should be completed with under *n* defects since our team has no prior experience with some facets of this project. Below is our quality plan for each iteration:

Iteration 1: Built-in Server, User registration/sign in/sign out

- Built-in Server: 1) Requirements reviews 2) design reviews 3) coding review 4) unit testing

- User registration: 1) Requirements reviews 2) design reviews 3) coding review 4) unit testing

- Built-in server + User registration: 1) Integration testing - w/ user account that has been created through registration 2) system testing - adding new accounts to see if the system is operating up to date 3) acceptance testing - see how the software manages and stores accounts

- Sign in/sign out: 1) Requirements reviews 2) design reviews 3) coding review 4) unit testing - evaluate UI 5) Integration testing - making sure login capabilities are synced with user registration 6) system testing - making sure the account properly changes states based on whether one is signed in 7) acceptance testing - software runs like any other account-based system

Iteration 2: User Profile Page, Trip Creation

User profile page: 1) Requirements reviews 2) design reviews 3) coding review 4) unit testing - test cases for the interface (what if I want to add this or that to my profile?) 5) integration testing - implementing the interface into the app 6) system testing - is the profile page running seamlessly with previously-built functionalities 7) acceptance testing - do we have a running user profile page

Trip Creation: 1) Requirements reviews 2) design reviews 3) coding review 4) unit testing - can i add a multitude of different kinds of trips with different attributes? 5) Integration testing - are trips properly connected to users 6) system testing - is this new component properly functioning within the scope of the software 7) acceptance testing - can trips be created as we would like them

Iteration 3: Google Maps API, trip scheduling/matching

Google Maps API: 1) Requirements reviews - are the requirements we plan to use possible with the API? 2) design reviews - Are we integrating maps and data from the API well 3) coding review - is the map doing what it is supposed to do on a broad scale 4) unit testing - test cases for the map esp. when dealing with dropping location pins and geolocating 5) Integration testing 6) system testing 7) acceptance testing

Trip scheduling/matching: 1) Requirements reviews 2) design reviews - have we implemented the right schematic for taking a created trip's location quality and attributing it to a geolocation 3) coding review - does our code optimally call on the google maps API we have tapped into 4) unit testing - test cases for different kinds of trips created

Google Maps API + trip scheduling/matching: 1) Integration testing - both features work sensically together 2) system testing - are two different trips within the same vicinity matching 3) acceptance testing - trip scheduling functions in the greater scope of the app itself

Iteration 4: Social network functionality: 1) Requirements reviews - how error prone are added functionalities 2) design reviews - do the designs we picked work within the framework of an individual profile? 3) coding review - is our code able to withstand the various test cases that it will undergo 4) unit testing - do the added functionalities work 5) integration testing - do any added functionalities have trouble on a profile-basis 6) system testing - the user experience of all features is smooth 7) acceptance testing - does the app feel complete?

4. Risk Management Plan

Risk Item	Probability	Loss	Cause?
Gold Plating	Low	Med	● Going down rabbit holes for non-fundamental features
Unrealistic Scheduling	Med	Med	● Challenging to accurately schedule our semesters far in advance
Productivity Issues	High	High	● We (Davidson students) all have busy schedule
Growth in/Unclear Requirements	Med	Low	● Adding new requirements to prototype iterations thorough consideration
Personnel Shortfalls (training/background)	High	Med	● From the offset the technology will be new to us

Three Most Critical Risks
Productivity Issues: To mitigate the productivity issues that may very well arise during the development process, our main strategy will be clear division of tasks. If we keep the tasks assigned to each person relatively short and well-defined, it will make it much easier for that individual to make progress and achieve the milestones necessary.

Personnel Shortfalls: To avoid major personnel shortfalls that may arise due to lack of training with the technology (Xcode/Swift/etc.), we will focus on productivity as we learn. The main necessity here is to be producing code for the project while working through tutorials and picking tutorials appropriately. For example, as we get used to the Swift language it would be more valuable to follow a tutorial outlining how to create a sign in/sign up page rather than an iOS game.

Unrealistic Scheduling: We hope that our process model selection will help us avoid running into problems with our inevitably busy schedules. It will also be important for all members of the team (not just the manager) to check in with each other's progress on tasks and possibly suggest scheduling adjustments when necessary.

5. Project Monitoring Plan
Measurements of the progression of our project will take the form of prototypes. Each prototype should have at least one major feature more than the previous and should be release roughly weekly in order to be completed on schedule. If not on track, we have

built in a buffer at the end of our schedule to allow for a couple of days on the end of a couple prototype's phases to finish up a feature that is taking longer than expected.

Within each prototype phase there will be informal checks on progress. This seems to work better on a project this small and with our student schedules. These will likely take the form of just a text message or e-mail to update other members on our progress a couple of times during the week.

6. Detailed Scheduling

Schedule	Days needed	Dates
Starting Date		Feb. 22
Create first user interface 9 (Labels, Text Fields, Buttons)	3	Feb. 22 - 24
Transition from login page to main app	3	Feb. 25 - 27
Milestone: make sure user will encounter no extraneous errors when inputting info.	1	Feb. 28
Individual information page, contains date, location, interests, etc.	5	Mar. 1 - Mar. 5
Milestone: test with our team members' information	1	Mar. 6
Team training: watch tutorials on how to create a server	5	Mar. 7 - March 11
Create a iOS server for our data	5	Mar. 12 - 18
Milestone: Upload our first ever user info	2	Mar. 19 - 20
Download any data from our server	4	Mar. 21 - 24
Sorting algorithm: match users by their interests	7	Mar. 25 - 31
Present matches to the user	9	Apr. 1 - 9
Send an email if matched, or other ways of contact	3	Apr. 10 - 12
(optional: chat box)		
Integration w/ Google Maps API	12	Apr. 13 - Apr.26
Add additional functions	6	Apr. 27 - May. 2
Final debug and testing	7	May. 3 - May. 9
End Date	**Total: 73 days**	May. 9