**Scope and Charter**

Real Time Bus Tracking App

University Of British Columbia Okanagan

COSC 499

Authors:

|  |  |  |
| --- | --- | --- |
| Document Owner(s) | Role | Contact |
| Wasek Habib | Product Manager | wasek.edu@gmail.com |
| Matthew De Leeuw | Integration Lead | mattdeleeuw@hotmail.com |
| Trevor Gallicano | DevOps Lead | trevorg@hotmail.ca |
| Ini Oladosu | Technical Lead | inioladosu@gmail.com |
| Kyle Rennie | Client Liaison | kyrenzie@gmail.com |

Document History:

|  |  |  |
| --- | --- | --- |
| Version | Date | Document Revision Description |
| 1.0 | October 23, 2018 | Initial document |
| 2.0 | November 20, 2018 | Updated to match changes |
| 3.0 | December 5, 2018 | Changes to requirements |

[**1. Identification**](#_t4hc7lj0qqd3) **4**

[**2. Project Purpose**](#_6brxl8u0h67a) **6**

[**3. Project Objectives**](#_dcxmj2fclgii) **6**

[**4. Success Criteria**](#_opazynbc209w) **6**

[**5. Terminology**](#_hsppcf42dlq5) **6**

[**6. Scope Statement**](#_r2kup6bybfym) **7**

[6.1. Out of Scope](#_4l2zw48vqhsp) 7

[**7. Requirements**](#_8umks4lr2rzk) **8**

[7.1. Functional Requirements](#_2iremv4tg5cg) 8

[7.2. Non-Functional Requirements](#_ouzpn13xryh8) 9

[Development](#_876qtgfyqquz) 9

[Performance](#_lkq2q1fjqt0) 9

[7.3. Technical Requirements](#_qe0ier6lbi44) 9

[7.4. User Requirements](#_a4x40uilqp2w) 11

[**8. Work Breakdown Structure**](#_vxrbbeli943v) **11**

[**9. Assumptions**](#_ncrhpjjgdfch) **13**

[**10. Environmental Constraints**](#_k4v2kyy0dqlb) **14**

[**11. Risks**](#_nvjbwcgg0irv) **14**

[**12. Project Development Methodology**](#_tc83umre5m8v) **16**

[**13. Project Milestones**](#_knhvkud45iwl) **16**

[**14. UML Use Case Diagram**](#_wqnbc2siz7lp) **18**

[**15. Approval**](#_61ndcoe204wg) **19**

# 1. Identification

Name of Project: Bus Tracking App

Sponsor: Fremtid Media

Project Stakeholders:

|  |  |
| --- | --- |
| **Stakeholder Names** | **Roles** |
| Fremtid Media | Sponsor/Client |
| Reza Afzali | Client |
| Simranpal Bains | Client |
| Scott Fazackerley | Instructor/Supervisor |
| UBCO Students/ Residents of Kelowna | Users |
| COSC Capstone Team 12 | Android App Developers |
| Engineering Capstone Team | Hardware/Infrastructure Team |
| University of British Columbia | Academic Institution |
| BC Transit | Open Data Provider |
| FirstCanada | Kelowna Public Bus Transportation Operator |
| Kontakt.io | Beacon Provider |
| Google | Android OS Provider |
| Here | Maps API Provider for Showing Bus Location on the App |

Team:

|  |  |  |
| --- | --- | --- |
| **Team Members** | **Roles** | **Responsibilities** |
| Wasek Habib | Product Manager | * Primarily responsible for the project workflow, requirements engineering, software architecture, and ensuring SDLC. * Involved in design and development process. |
| Ini Oladosu | Technical Lead | * Responsible for the non coding documents * Responsible for UI Design of application * Involved in development process |
| Matthew De Leeuw | Integration Lead | * Responsible for checking code before integration into master branch (aka) responsible for what gets deployed * Responsible for programming with team on seperate features |
| Trevor Gallicano | DevOps Lead | * Responsible for continuous integration/deployment * Responsible for automated testing * Involved in development process |
| Kyle Rennie | Client Liaison | * Responsible for continuous client contact and logging conversation * Involved in development process |

# 2. Project Purpose

Currently there is a ridership of 4.9 million in Kelowna per year, a fair amount of riders (>1%) use the transit app, with this app we are looking to take a portion of the demand for maps and transit for this app. Current apps have issues with telling the user the real time of their bus arrival because they are unreliably crowdsourced, or just use the bus schedule for times. The Transit App has a “GO” function but this is not always available. Bus drivers also want to maximize efficiency but they don't have a way to know if passengers are at the upcoming stops. Another problem transit users have is that not every bus stop has audible notifications. The purpose of the project is to provide a reliable solution to the commuters for tracking their desired busses in real time, passengers waiting at the bus stop to be picked up with an android application, and an audio notification of the upcoming bus to the people waiting at the bus stop. If the project is not completed on time then the company overseeing it can chose to continue with a new or the same team. They would have prototypes and documentation to help them continue with the project, even with a new team.

# 3. Project Objectives

Create a solution for commuters to track their desired bus in real time using an android app. Passengers waiting at a bus stop should be confident that the bus they are waiting for will pick them up. Passengers waiting at a bus stop who are not paying attention or maybe are visually impaired can have audio notifications from a speaker on the bus stop. A light sensor on the bus will notify the bus driver, if there’s a passenger waiting at the bus stop and all the passengers have boarded.

# 4. Success Criteria

* The app will have simple and elegant user experience following standard human computer interaction rules.
* User satisfaction will be measured by a standardized set of questions followed by the usability test and will measure 3 or higher on a 5 point scale.
* The user interface will follow the best UI practices and ‘material design’ standards.

# 5. Terminology

* Beacon: Small transmitters that use Bluetooth to send a single signal containing small data packet for other devices to pick up.
* REST API (Application Programming Interface): Tool that helps to communicate between two programs using HTTP protocols.

## 

# 6. Scope Statement

The project has two parts: Hardware and Software. COSC team is responsible for anything related to the server, development and deployment of the android application and its connection to the beacons (software side). The android app (proof of concept) will show the user live location and arrival time of the bus (real-time bus schedule) in nearby bus stops. The user will get a push notification of the remaining arrival time of the subscribed bus number. The beacons will be installed and configured on a bus/car and at bus stops by the engineering team. The engineering team is also responsible for small single board computer on the bus. The cloud server will send information (bus stop location) of passenger tracking a bus to internet enabled small board computers. The computers will then communicate with the light sensors and speaker and make some notification. (done by the engineers). If the user clicks on the push notification, it will take him/her to the app and show the live location of the bus on the map. The users will not need an account to use the application. Personal information of the users will not be asked or stored. However, their travel time and distance will be stored in a database.

6.1. Out of Scope

* The engineering team is responsible for setting up and configuring beacons and programming small single board computers.
* Receiving data from the server to the small board computers and communication between small computers and speakers and light sensors are also EE team’s responsibility.
* Range of beacon.
* Accuracy of location tracking will not meet any standard
* Data protection and transmission security of the beacons. Kontakt.io is responsible for secured data transmission of the beacons.
* The coverage area will not be outside of UBCO roundabout and EME bus stop.
* Setting up the beacons, checking physical conditions and health.
* Creating mesh network.
* People with strollers or wheelchairs (physically impaired).
* Setting up and working with speakers and displays on bus-stops.
* Any communication between speakers, beacons, and light sensors.
* Any kind of hardware implementation, security, and maintenance, and programming the small board computers are done by the engineering team.
* Cost analysis.
* Wayfinding (Navigation and in-between stops).

# 7. Requirements

## 7.1. Functional Requirements

* Android Application will ask for user’s permission to access location, and to store information. --
* Android Application will display a map with the user location. --
* App will let the user to track a bus/car.
* Android Application will display a map with the tracked bus location.
* Application will display the expected arrival times of the selected bus.
* Application will receive information from the nearby beacons.--
* Application will send nearby beacon information to the server.-
* User will receive a push notification when the selected bus has arrived at the stop.-
  + Notification sent to phone from AWS
* Application will be available to all users with no registration.--
* Application will show a list of nearby bus/car that has/have LTE enabled small board computer with GPS tracker board.
* App will track travel distances and travel times of a user.
* Admins will be able to view data stored in the online database hosted in AWS.
* Cloud server will get user location when he/she is in the beacon range of the bus location.
* Cloud server will receive beacon information from the application.-
* Cloud server will receive user selected bus stop and time reminder from the application.
* Cloud server will perform some computations to check if the user is on the bus.
* Cloud server will perform some computations to check if the user is out of the bus.
* Cloud server will send tracked bus location to an LTE enabled computer at bus stop/bus.
* Cloud server will store user’s travel time anonymously in database.
* Cloud server will store user’s travel distance anonymously in database.
* Cloud server will send tracked bus location to the application.
* Cloud server will send bus stop location of the passenger to an LTE enabled computer on the bus.
* Cloud server will get data of the live position of busses from small board computers.

## 7.2. Non-Functional Requirements

### Development

* Developed as an Android application written in java.
* Development process will be completed using the agile method.
* Application will use third-party API’s.
* Costs for development (such as cloud service or third party API’s) will be covered by the client.

### Performance

* User must be connected to the internet.
* Software will handle at least 2000 users.
* Software will be scalable up to the entire 97 bus route in Kelowna.
* Bus position will be updated within 3 seconds (including screen latency).
* Application and small board computers use http calls (REST APIs) to communicate with the server.
* Bus tracking will only show information related to the next bus(es).
* App will not track location in the background

## 7.3. Technical Requirements

|  |  |
| --- | --- |
| **Tools/Languages/Libraries** | **Type** |
| Android Studio | Development Environment |
| Balsamiq | Wireframing |
| Adobe XD | Mockup |
| Docker | Container based software as a service |
| Jenkins/Travis | CI Solution |
| Git | Version Control System |
| Github | Version Control Hosting Service |
| Trello | Project Management |
| Slack | Communication |
| SQLite | Relational Database |
| Couchbase Mobile | NoSQL database |
| AWS/Firebase | Cloud Platform |
| Toggl | Time Tracking |
| Java | Client side Programming Language |
| Node.js | Backend Programming Language |
| Postman | API debugging and testing tool |
| Swagger | API designing and documentation tool |
| Here Maps/Google Maps | Maps API |

* The application will connect to the beacon mesh network using Kontakt Beacon APIs. API key will be stored in the app.
* The application will display a map with the bus locations using Here maps API and caching the latest location. API key will be stored in the app.
* Application will ask for user’s permission to access to GPS, bluetooth, and internet using AndroidManifest.xml and java file to ask for user permission.
* System will track the live position of busses using GPS location from small embedded computer on the bus.
* Application will display the expected arrival times of bus using GPS location from small embedded computer on the bus and Here maps APIs. API keys will be stored in the app. Beacon UUIDs and geo-coordinates will be stored in the database.
* User will receive a push notification when they are within a set distance of a bus stop beacon using async push notification. User location and desired bus location in the background will be tracked using GPS and Kontakt Beacon APIs.
* Application will show a list of nearby bus/car(s) that has/have beacon installed using Kontakt Beacon APIs and BLE. API keys will be stored in the app.
* App will only receive info from beacon
* App will track the current location of a user in the background using GPS.
* App will track travel distances and travel times of a user using Here maps APIs to calculate the distance and time. API keys will be stored in the app.
* Travel data will be sent to an online database when the device has an online connection using /POST, /PATCH, /GET and /DELETE APIs.

## 7.4. User Requirements

* Use application without an account.
* User receives push notification when entering specified beacon range.
* Users can track bus with map.
* User can see expected bus arrival time.
* Can not log in.
* Able to track single bus.
* Able to use application on Android device.
* Users are Identified by Instance ID, or mac address. This ID does not change.

# 8. Work Breakdown Structure

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Task List | Owner(s) | Estimated Hours |  |  |  |  | Total Estimate |
|  |  |  |  |  |  |  | 186.7 |
|  |  |  |  |  |  |  | Average Estimate |
| **Learning** | All | Ini | Kyle | Matt | Trevor | Wasek | 41.6 |
| Android | ✔ | 17 | 15 | 20 | 12 | 10 | 14.8 |
| Maps | ✔ | 3 | 2 | 3 | 3 | 5 | 3.2 |
| Kontakt.io | ✔ | 5 | 3 | 3 | 3 | 5 | 3.8 |
| CI Tools | ✔ | 12 | 2 | 12 | 12 | 12 | 10 |
| AWS/Firebase | ✔ | 5 | 2 | 4 | 2 | 3 | 3.2 |
| Databases | ✔ | 2 | 2 | 4 | 2 | 3 | 2.6 |
| Node.js | ✔ | 5 | 5 | 4 | 4 | 2 | 4 |
|  |  |  |  |  |  |  |  |
| **Meetings** |  | 36 | 36 | 36 | 36 | 36 | 36 |
|  |  |  |  |  |  |  |  |
| **Design** | All |  |  |  |  |  | 13.6 |
| Documentation | ✔ | 5 | 15 | 4 | 5 | 16 | 9 |
| Paper Prototype | ✔ | 2 | 4 | 2 | 1 | 1 | 2 |
| WireFraming | ✔ | 2 | 2 | 1 | 1 | 2 | 1.8 |
| MockUp | ✔ | 3 | 2 | 2 | 1 | 1 | 1.8 |
|  |  |  |  |  |  |  |  |
| **Development** |  |  |  |  |  |  | 79.5 |
| ***1. Maps API*** | Kyle |  |  |  |  |  | 10.8 |
| 1.1 Integrate Maps Into GUI | ✔ | 5 | 3 | 8 | 3 | 1 | 4.8 |
| 1.2 Show Where Bus is Located |  | 5 | 3 | 2 | 4 | 5 | 3.8 |
| ***2. Kontakt Beacon API*** | Matt |  |  |  |  |  | 3 |
| 2.1get location data from beacons, or arduinos. | ✔ | 4 | 4 | 3 | 3 | 1 | 3 |
| ***3. Cloud Service*** | Wasek |  |  |  |  |  | 7.5 |
| 3.1 Receive user and bus locations | ✔ |  |  | 3 | 2 |  | 2.5 |
| 3.2 Compute bus stop location relative to bus |  |  |  | 3 | 2 |  | 2.5 |
| 3.3 Send bus location to app |  |  |  | 3 | 2 |  | 2.5 |
| ***4. Android SDK*** | Tbd |  |  |  |  |  | 8.4 |
| 4.1 Collect GPS Data | ✔ | 3 | 3 | 2 | 2 | 2 | 2.4 |
| 4.2 Store GPS Data |  | 5 | 2 | 2 | 3 | 8 | 4 |
| 4.3 Permissions | ✔ |  |  | 1 | 1 |  | 1 |
| 4.4 Push Notification | ✔ |  |  | 2 | 2 |  | 2 |
| ***5. Build API*** | Wasek |  |  |  |  |  | 13.4 |
| 5.1 Integrate Android with Cloud |  | 4 | 4 | 4 | 4 | 4 | 4 |
| 5.2 Local Database to Cloud Database Path | ✔ | 15 | 3 | 10 | 3 | 16 | 9.4 |
| 5.3 Integrate beacons to cloud |  |  |  | 1 | 5 |  | 3 |
| ***6. User Interface*** | Ini |  |  |  |  |  | 6 |
| *6.1 Bus Number Button* |  |  |  | 1 | 2 |  | 1.5 |
| *6.2 Track Button* |  |  |  | 1 | 2 |  | 1.5 |
| *6.3 Bus Selection Button* |  |  |  | 1 | 2 |  | 1.5 |
| *6.4 Bus List uses recycler list* |  |  |  | 1 | 2 |  | 1.5 |
| *6.5 Display expected arrival time of bus* |  |  |  | ??? | 3 |  |  |
| **7. Integration and Testing** | Trevor |  |  |  |  |  | 32.4 |
| Ci Tools Setup |  | 4 | 1 | 3 | 6 | 1 | 3 |
| Usability Testing |  | 4 | 2 | 5 | 8 | 5 | 4.8 |
| Documentation | ✔ | 5 | 3 | 3 | 6 | 16 | 6.6 |
| Unit Testing |  | 2 | 2 | 8 | 10 | 10 | 6.4 |
| Integration Testing |  | 5 | 4 | 8 | 8 | 8 | 6.6 |
| Acceptance Testing |  | 5 | 5 | 5 | 5 | 5 | 5 |
| **Final Documentation and Presentation** |  | 16 | 16 | 16 | 16 | 16 | 16 |

# 9. Assumptions

# UBCO will approve installing beacons on campus.

* A beacon will be set up in one bus upon approval from BC transit. If not, the students will use their own transports. The engineering team is responsible for this.
* The engineering team will be responsible for installing and configuring beacons and will provide any help needed as soon as possible when COSC team test them. Both team will be present during the integration testing.
* The beacons given will work and respond as expected. The client and engineering team are responsible for any faulty beacons and hardware issues.
* The developers have knowledge of Git, Java and relational databases.
* Third party APIs (i.e. maps API, beacon API) will work properly.
* The team built APIs will be public and will not ask for any authentications.
* The cloud service (AWS/Firebase) will have at least 99.5% uptime.
* The beacon range will be 50-70 meters.
* The beacons will not be stolen or vandalized.
* The team will be agile, self-organized, and maintain synergy.
* Any expenditure including cloud service, third party APIs, transportation logistics will be provided by the client.
* The communication between stakeholders will be prompt, clear, and efficient.
* Each developer will be working a minimum of 8 hours during school weeks.
* There will not be any requirement changes from the client after October 16, 2018.
* The Beacon response time will be 1-2 seconds.

# 10. Environmental Constraints

* Requires approval from UBCO and BC transit to install beacons on campus and bus.
* The durability and performance of the beacons in Kelowna weather is unknown.
* Limited in-person meeting with the client.
* Field testing will be challenging during hostile weather.
* Development tool licenses and term of service.
* No control over third party APIs and cloud service.
* Some tasks are dependent on the task completion of engineering team and can not be done in parallel.
* Majority of the developers are not familiar with Android ecosystem, building REST APIs with node.js, and NoSQL databases.
* Lack of domain knowledge will lead to unrealistic estimations.
* Time constraint.

# 11. Risks

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Likelihood | Severity | Mitigation Plan |
| Unfinished Project | Medium | High | Plan to stay ahead of deliverables. |
| Conflict with Engineering Team | Low | High | Frequent communication and transparency. |
| Losing members | Low | Medium | Involve all team members in development so we can pick up where other left off. |
| Unmotivated Team Members | Low | High | Constantly staying ahead of deliverable due dates |
| Faulty Beacons | Medium | High | Have as many as possible so we are not reliant on just a few. |
| Client company shut down | Medium | High | Continue project with Scott. |
| No Control Over 3rd Party APIs | High | Medium | Study the APIs as much as we can so we call deal with any issues if they arise. |
| Environmental Constraints | Medium | High | Have backup plans, which will be discussed with the client. |
| Beacon Security | Medium | High | Change beacon provider or implement own security. |
| API Security | Low | High | Implement API key and authentication |
| Device Security | Low | Low | Delete device database as soon as possible. |

## 

# 12. Project Development Methodology

Scrumban methodology and Trello will be used for project management. The sprints will be one week long starting from every Tuesday. Weekly sync up, demo, planning meeting will be held on Monday. If Monday doesn’t work out for some reason, the team will have meeting on Friday or Tuesday before the class (not recommended). Backlog will be created in each week’s planning based on the progress and priority. Every week the developers will pick up the tasks from backlog they think they can finish in that week. If a task seems big, sub tasks can be created under that task. The tasks will have story points. 1 point represents 1 hour. Everyone is expected to finish at least 80% of their workload each week, although 100% is preferred. The rest can be carried forward to the next week’s sprint. There is no need for daily scrum meeting as trello is synced to the scrum channel. So everything knows who is working on what.

The developers will create and push commits to the branches named with task numbers and create pull requests. Only integration lead or feature owners can review code and merge pull requests to the “Development” branch. The integration or devOps Lead will then update the master branch. The code will be reviewed after running automated test cases and a successful Jenkins build. Non-code documents will be merged by the technical lead.

# 13. Project Milestones

|  |  |  |
| --- | --- | --- |
| Milestones / Deliverables | Due Date | Complete (YES / NO) |
| Scope and Charter Document & Presentation | October 23th, 2018 | YES |
| Design Requirements Document & Presentation | November 13th, 2019 | YES |
| Integration Testing with the Engineering team | November 16th, 2019 | NO |
| MVP Presentations | January 8th, 2019 | NO |
| Testing Documentation and Presentation | TBA | NO |
| Final Deliverable | TBA | NO |
| Final Project Presentation | April 8-26 | NO |

# 14. UML Use Case Diagram

## 

# 

# 

# 

# 

# 15. Approval

Product Manager:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Sponsor:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_