October 28, 2017

Project Plan

TRAVLENDAR

Prepared By: Group 2

Aditya Nimbalkar

Alfred Gonsalves

Deepak Haldar

Mehul Shah

Melroy Dmello

Param Mehta

Table of Contents

Project Description

Overview (CO-5):

- Travlendar will provide flexible and fully-featured calendar support that considers the travel time between meetings.
- The goal of this project is to create a calendar interface that automatically computes and accounts for travel time between appointments to make sure that you're never late for an appointment. This project will ensure that users will be notified in case they schedule conflicting meetings. The project will also send notifications informing users when they should leave for the next meeting.
- The users for this application include people who work "on the go", travelling from meeting to meeting throughout the day as well as busy parents.

Key Requirements (CO-5):

- The application should compute travel time between meetings and prevent scheduling meetings whose start/end times conflict as a result of inability to travel.
- Suggest the best option of travel between each meeting.
- Create overall compatible schedules, considering the travel mode preference of the user.
- The application should send notifications reminding a user of the next meeting. The travel time should be taken into consideration for sending the notification at the appropriate time
- On a given day, a user's car (or bicycle, skateboard,etc) should start and end at the user's home.

Deliverables (CO-5, CO-6):

- Project Plan Document
- Architecture Document
- User Manual
- Hosted Web Application
- Mobile Application (Android/iOS compatible)
- Code (Version Control and Bug tracking)
- Domain Name and the credentials for the associated with the domain.

Acronyms and abbreviations (CO-7):

CRUD : Create Read Update Delete

Design and Architecture (CO-1, CO-3)

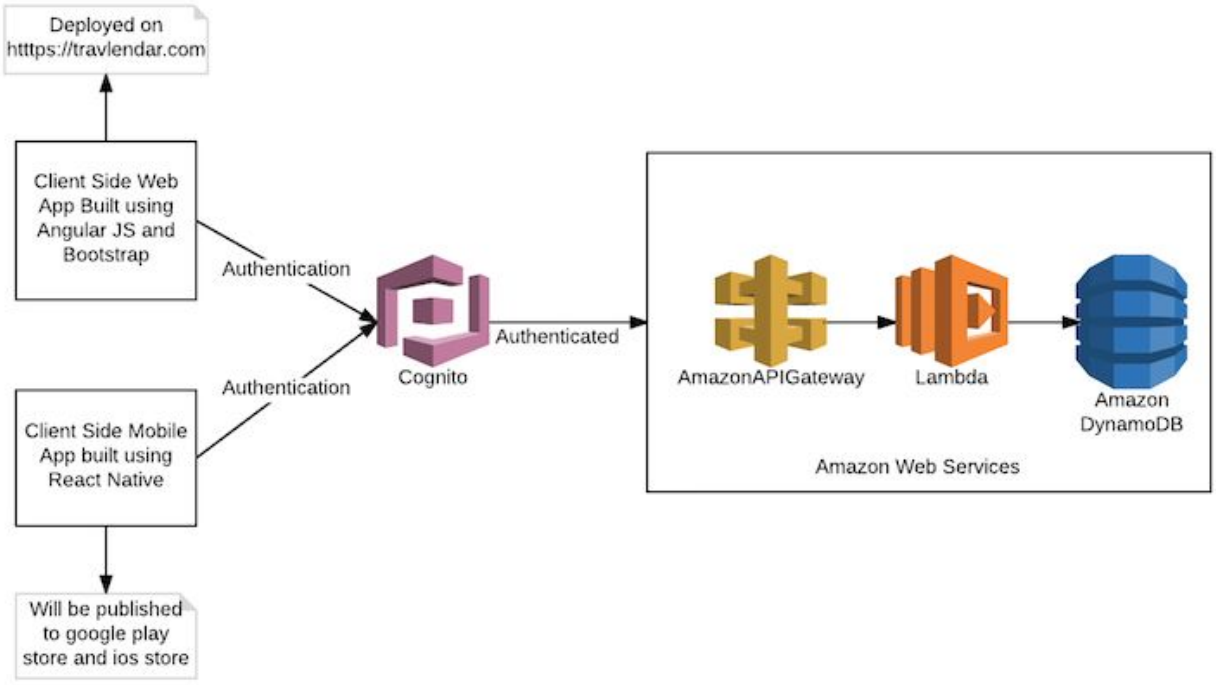


Fig 1.0 Architecture of TravLendar

- The Architecture of Travlendar is based on serverless architecture described here
- The front end of the application will be built using Angular JS, Bootstrap and SCSS as the main technologies along with other plugins. The backend of the application will be developed in Amazon Web Services. The Specific AWS services this project will incorporate are :
    - Cognito : To handle user registration, authentication and social identity integration
    - API Gateway : To expose a Secure REST API which *Authorized* clients can connect to
    - Lambda : Contains the core logic for the application along with algorithms.
    - Dynamodb : Used to persist the data
    - SNS : Used for sending app notifications
    - SES : Used for sending Emails
    - S3 : Used for holding the client side code which will be hosted
    - Route 53 : Used as DNS (our site is currently live on https://travlendar.com)
- There will be a common backend exposed by API's which will be used by the Web Application and Mobile App.
- Since a calendar interface is not very responsive and will not scale well from desktop size devices to mobile devices, a separate mobile app will be developed using React Native which will consume the backend API's

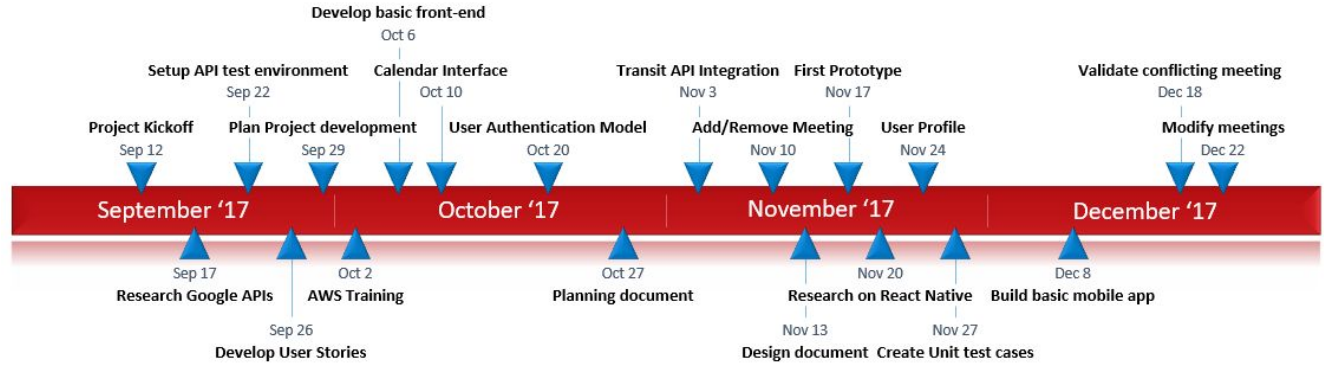
Implementation Strategy

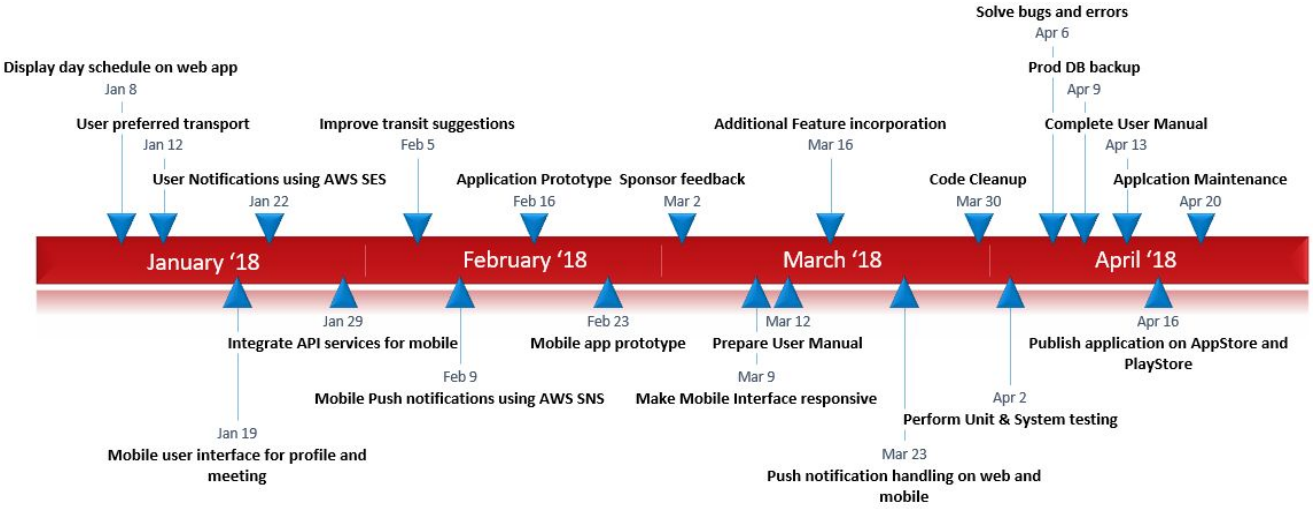
High-level Work Breakdown Structure (CO-2):

Task	Description	Skillset Required	Expected Time
Setup Development Environment	This includes setting up tools, IDE and server setup. Installing SSL certification on the server and working environment. Creating AWS account for backend and Google account for API Integration	Knowledge about the technology stack of AWS, REST Architecture.	2 weeks
User Authentication Module	Create the login or register page to use the application, also added the feature to use social accounts to sign up into the application.	AngularJs, AWS Cognito	1 week
Schedule meeting	Able to Schedule a new meeting, Read the Meetings present in the calendar, Edit or remove the Meetings present in the system.	Angular Js, AWS lambda, Dynamo DB	3 weeks
User Notification	Sending notifications to the user about the next meeting or time to leave for next meeting through email used to sign up for the application.	AWS SES	1 week
Mobile Development	On the same lines, create a React native application, which can be used cross-platform for mobile devices supporting using Android and iOS	React Native JS	6 weeks
Push Notification	The push notification feature on the user's mobile device from the application installed. Incorporate Handling of app after the push notification is clicked by the user	AWS SNS	2 weeks
Suggestion in Transit travel time	The transit time for the meeting from previous meeting's location to the next meeting.	Angular JS, AWS Lambda, Dynamo DB	1 week
User Preference and Setting	The settings page where user selects the preference at what time during the day he wants notification for entire schedule of the day.	Angular JS, Dynamo DB, AWS Lambda.	2 weeks
Incorporate User	Use the work address, home address, lunchtime, maximum walk distance for a day	Angular JS, Dynamo DB, AWS	3 weeks

Preferences and settings on meeting scheduling	and other settings that can be found under profile page in the application.	Lambda.	
Unit Testing and System Integration Testing	Testing each and every scenarios present in the application rigorously.	Taiga, Firebug.	3 weeks
User Manual	The document indicates application features and how the application can be used by the user.	Microsoft Word, Adobe PDF.	1 week

Schedule / Timeline (CO-2):





REQUIRED HARDWARE (CO-2):

- The project does not require any special hardware other than a mobile device with Android/iOS compatibility or a laptop/ desktop PC with a browser to open the link to the web application.

THIRD PARTY CONTENT (CO-2):

- All 3rd party libraries being used in the project are open source libraries covered by the MIT License.
- The main plugin of the application is the calendar plugin found here. This Plugin is also covered by the MIT License.
- All the place information and distance matrix between places are captured using Google API services

Quality (CO-2):

Our end code will meet the following quality requirements

- The front end code will adhere to the best practises of the published style guide of AngularJS found here.
- The code itself will not contain any errors found by the static code analyser ESLint
- Test cases will be developed for each module which will generate reports to get a quantifiable measure of the quality of code.
- The code shall be well commented and JSDocs for the same shall be maintained to make the code maintainable.

Other Special considerations (CO-7, Co-3):

- None

PROCESS

Process Description and justification (Co-2)

Agile Methodology is followed in the implementation of project. Each task is broken down into user stories. These user stories can be any features required in the product, bug fixes or any non-technical requirements. Every user story has title and description which helps developer understanding the task to be performed.

We follow a two weeks sprint cycle. We are using Taiga software to follow Agile Methodology. All the user stories are stored in Taiga which acts as the Product Backlog for the project. Depending on the project status, few user stories having critical importance or priority are selected in the sprint. Each user story is assigned to a team member who provides weekly status regarding its completion to other team members and also informs about any difficulties encountered in finishing the task

The Sponsor has always focused more on implementation and development of project and placed less importance on documentation. This made us decide to proceed with Agile Methodology. Also, the requirement specification is not constant as the sponsor has suggested and requested optional features in addition to the initially mentioned features. We report our progress to the sponsor on a biweekly basis through video conference and incorporate any suggestions or requested features by adding them to the Product Backlog.

Tools (CO-2):

- WebStorm - Used for https://www.jetbrains.com/webstorm/
- Adobe Brackets - http://brackets.io/
- Git - https://git-for-windows.github.io/
- Slack - https://slack.com/
- Taiga - https://taiga.io/
- AWS CLI - https://aws.amazon.com/cli/

ROLES AND RESPONSIBILITIES (CO-2):

USER	ROLE	RESPONSIBILITIES
Christine Julien	Project Sponsor	Approve deliverables, provide high-level direction.
Param Mehta	Front-End Developer	Design the UI for Calendar, the meeting scheduler and the Settings page for the user.
	Spokesperson	Write the meeting minutes for the team meetings and keep the coordination with Professor, Team and the Sponsor.
Deepak Haldar	Front-End Developer	Design the UI for Calendar and Profile Page for the user.
	Scrum Leader	Manage each development sprint and the daily Scrum.
Mehul Shah	Back – End Developer	Develop conflicting meeting and scheduling meeting functionality on the backend. Created POC on the distance matrix API
	Test Engineer	Creating unit test cases and validating the feature before integrating with the system
Melroy Dmello	Full Stack Developer	Develop The authentication module, notification module and schedule meeting functionality end-to-end,
	Project Co-ordinator	Assign tasks and maintain quality of code

Aditya Nimbalkar	Mobile Application Developer	Design the calendar component, authentication module and Schedule meeting module for mobile application.
Alfred Gonsalves	Back – End Developer Front-end integration	Develop delete functionality for scheduling the meeting. Integrated transit API on the front end and work on optimal routes for multiple meetings.

Location of Project Artifacts (CO-2):

- The code for the project will be hosted on GitHub (click here)
- The necessary documents will also be hosted in the same Github repo under Documents folder

Sponsor communications (CO-7):

- Communication with the sponsor will be on a bi-weekly basis using google meet.
- Prompt communications and ad-hoc queries which don't require a meeting are communicated via mail.

Risk management

identified Potential risks (CO-2):

Risk	Expected Incidence Rate	Impact
Technology does not meet Specification	10%	1
Late Delivery	10%	2
End Users Resist Application	20%	2
Changes in Requirement	20%	2
Lack of Development Experience	30%	2

Database is not stable	20%	1
Poor Quality Documentation	20%	3
Deviation from Software Engineering Standards	5%	2
Poor comments in Code	25%	4
Lack of Code Reuse	10%	4

Impact Values:

1 - Catastrophic

2 - Critical

3 - Marginal

4 - Negligible

MITIGATION STRATEGIES (CO-2, CO-3):

Risk	Mitigation Strategy	Cost of mitigation (for currently Active Strategies)
Technology does not meet Specification	Routine Meetings need to be conducted to ensure that the developers' idea of the application does not differ from the Sponsor's.	-
Late Delivery	Measures need to be adapted to ensure appropriate delivery per sprint in accordance with the scope of the Project and the submission deadlines.	-

End Users Resist Application	Keeping the End Users in mind, the User Interface needs to kept as simple as possible. Additionally, opinions from all stakeholders involved and results from surveys conducted to that effect need to taken into consideration.	-
Changes in Requirement	Routine meetings should be conducted so that the developers are notified of any requirement changes in a timely manner.	-
Lack of Development Experience	The entire development team will be required to learn and understand all technologies used in development. The member of the team most experienced in a particular field should instruct the others.	The Timeline gets affected due to the additional time consumed for the learning phase.
Database is not stable	The back-end developers should monitor any code that interacts with the database for defects or errors. Any issue should be brought to the notice of the entire development team.	-
Poor Quality Documentation	Regular meetings should be held to discuss all documentation topics and suggestions.	-

Deviation from Software Engineering Standards	Technical reviews need to be conducted on a regular basis.	-
Poor comments in Code	The developers should try to adapt to formal coding conventions and improve comments in code to boost understanding of the code.	-
Lack of Code Reuse	The practice of code reuse should be encouraged in order to minimize development time.	-