Watchnode

ITP_WE_B01_09

Malabe

Title:	Watchnode	
Batch :	Weekend	Group No: 1
Tech Stack :	MERN Stack	

	Surname with initials	Registration Number	Contact Phone Number	Email
1.	Perera G.D.A.K.	IT21072642	0763410607	it21072642@my.sliit.lk
2.	Fernando W.P.A.M	IT21078378	0764060737	it21078378@my.sliit.lk
3.	Jayawickrama K.O.	IT21086366	0766577545	it21086366@my.sliit.lk

Contents

Background

Schedules are a common part of day-to-day life. They help us get organized and are designed to make sure that we do not miss out on important events. These events could vary and could be even as simple as a normal cleaning task marked as needed to be done at a specific time. However, for those which aren't simple or frequent as the earlier it's a usual practice to record it somewhere maybe on a calendar physically or even digitally on a computer or mobile device which will act as a reminder that there's something which needs attention. Certain schedules may already be handed out to us and majority of the time it might be on a paper booklet or a digital copy. For example, a school timetable could be printed and handed out on a piece of paper and a schedule for a series of online webinars could be made available on a website as an image or a pdf document but ultimately, whatever the case, nine out of ten times it's up to us to mark the events in those schedules or register alarms for them, just as fail safes to act as reminders if we are to ever forget them.

Problem and Motivation

While it may seem fair and simple enough at first, when things get too busy or compact or when the process needs to be carried out for multiple schedules, it could very well turn out to be a tedious and time-consuming task which may be utilized for something more important. Without a proper reminder system, it is possible for events to just slide past us without being noticed or even if noticed it could be too close to date and we might not have the necessary time to prep ourselves for it, a scenario which is no less painful than the former. Schedules might even conflict with each other which worsens the question in case.

A successful solution to this problem may prove to be a lifesaver, taking quite a weight off one's shoulder of needing to keep track of each and everything in their day-to-day schedule. It may be an undisputed saver of time and energy acting more or less like a digital assistant passing out information on what's to come.

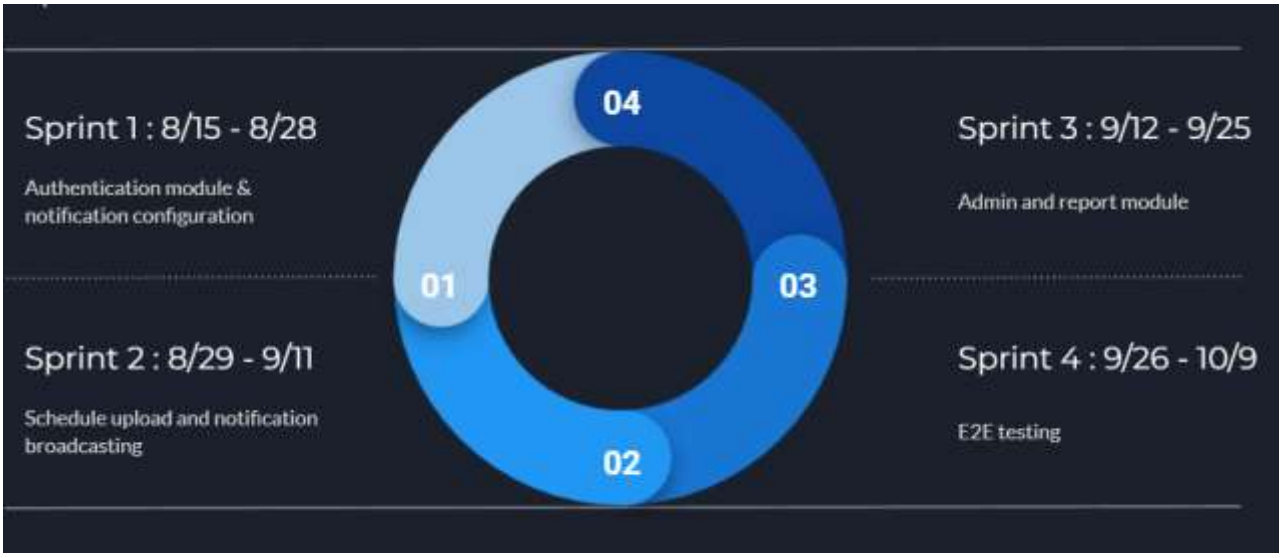
Aim and Objectives

Watchnode is an IT based solution which would provide the necessary cover to counter this problem. The outcome would be a web platform and mobile application which would allow a user to upload an image of a schedule which they already possess, a school or lecture timetable for example if we go by earlier discussions and the end result would be a digital

calendar for that schedule upon the analysis of which alerts would be broadcasted to the user at pre-configured time intervals facilitating reminders for the events in the uploaded schedule. Both the web and mobile applications will consist of the same functionalities where the only difference would be that the mobile application would facilitate an offline operations system just in case the user wishes to make use of the system without any internet connectivity.

The target would be to market the solution as fast as possible as a MVP release. The audience would include anyone who's lazy enough to maintain their own calendars or those who just do not have the time nor energy to do so. The system would be open sourced with a GNU General Public License and would welcome contributions from any third-party entity.

The development of the system will follow four sprints spanned out through a period of 8 weeks with each responsible for a key deliverable in the system.



Sprint 1: 8 / 15 – 8 / 28

- Will be covering the complete end to end authentication functionality of the system and notification setting configuration

Sprint 2: 8 /29 – 9 / 11

- Will be covering the schedule uploading and notification broadcasting

Sprint 3: 9 / 12 – 9 / 25

- Will be covering the administrator module and report module

Sprint 4: 9 / 26 – 10 / 09

- Will be covering an end-to-end test of the system

The modules targeted by each sprint would be covered in detail further down in this report.

Git Branching

Watchnode will use Git for its version controlling and as such all repositories will follow the following branching strategy.

Master

- Primary branch of the repository aligned with the main stable release of the system at any given point in time

Development

- Secondary branch of the repository containing the latest development code changes made by developers at any point in time

Feature branches

- Feature based branches broken from development and contain isolated codes related to an individual feature development. These branches will be merged back into development once the development is complete

Fix / Hotfix branches

- Branches containing fixes and hotfixes

At the end of each sprint the latest changes from development branch will be merged with master and aligned with the stable release.

System Overview

Due to being an extremely simple system the functional requirements of Watchnode could be categorized into 6 different modules. Out of these modules the priority for each is evaluated based on the range of 1 to 3 where 1 would denote the highest priority

- Authentication
- Priority – 1
- Description

    - Authentication will start with the process of user registration upon which the user will receive a confirmation mail to activate his account. On successful confirmation of an account the user would be able to login to the platform using the credentials specified during the registration stage

    - In case a user is to forget his/her password Watchnode will allow it to be reset where an email will be sent to the registered email of the user with a link which could be visited and followed through to successfully reset the password

- Technical specifications

    - Authentication module will facilitate two user roles, ADMIN and USER for use by normal users and admins of the system. An admin upon successful login will be provided with additional management features of the system through a separate dashboard visible to only ADMIN user roles

    - Authentication strategy will be JWT where upon each successful login the server will generate a JWT access token and refresh token and send it along with the response. This token would be saved on the client side and passed back to the server with each subsequent request

    - Encryption strategy to be followed will be Bcrypt where the password would be hashed from the frontend and passed to the backend where it would be compared with the actual user's password

- Schedule Management
- Priority – 1
- Description

    - User will be provided with the option to upload a copy of his / her schedule to the system. It could be either an image, pdf or excel file

    - Upon successful upload of the schedule the user would be presented with a confirmation view of the generated calendar. To handle situations where the schedule analysis was not a 100% accurate the user will be provided with the option to manually edit the digital output before confirmation

    - User will be able to manage previously uploaded schedules (edit or delete them which will remove their related events from the calendar). This also means that the user will be able to upload and maintain multiple schedules.

- Technical specifications

    - The tesseract OCR engine developed by google will facilitate the schedule analysis. It supports over 100 languages and is available for node through the npm wrapper node-tesseract-ocr and for browsers though the library tesseract.js

    - Attention will be paid to the analysis process to ensure speed and efficiency of the scanning and to the security of the uploaded data (Data uploaded by one user will never be visible to another user not even an admin)

- Notification settings configuration
- Priority – 1
- Description

    - Users will be able to configure the time periods based on which they receiver alerts and whether the alerts are enabled or disabled. The setting should be available at a global level and at individual levels for a schedule or an event inside a schedule

    - Users will be able to choose the type of notification which they receiver whether it's an email or a push notification

    - Security will apply to this as well like the previous functionality, a user's settings will be able to be modified by that user only.

- Notification broadcasting
- Priority – 1
- Description

    - The system will analyze user schedules periodically and broadcast notifications based on their configured alerts

- Technical specifications

    - The system will make use of firebase cloud messaging for push notifications and Gmail for email transmissions. Gmail was chosen over another service which was looked at which was SendGrid and was ultimately picked on the plate due to its increased email quota in the free tier which is 500 / day in contrast to 100 / day respectively. SendGrid however provides a list of open-source custom email templates which will be leveraged and edited to customize Watchnode.

    - In the case of push notifications support will be provided in the system to push notifications to the same user even if he / she is logged in from multiple devices

    - The notification broadcasting would be implemented by a cron job which even though is not the cleanest solution, will be the fastest way to market. In contrast to this serverless would be a better approach.

- Speed is of the utmost important over here and steps should be taken to minimize any delays in the notification broadcasting

---

- Admin module
- Priority – 2
- Description
    - Admins will be provided with a separate dashboard used for monitoring and a separate section for admin user management
    - Admins will be added through this user management module and would receive a welcome email with an auto generated password
    - Upon a successful login to the system an admin will be focused to a compulsory password reset
    - An admin will have the capability to edit or remove another admin from the system where in the case of edit, email editing will be disabled and if it's a removal in question an email will be sent to the removed user notifying of the event
- Technical specifications
    - The dashboard API queries will be thoroughly analyzed for efficiency and optimized for speed as they will be dealing with huge data sets

---

- Report generation
- Priority – 3
- Description
    - User will be able to generate a list of all upcoming notifications that he/she has subscribed to and all those which have already been broadcasted

    - From an admin's point of view there will the functionality to see a list of all notifications sent out by the system with filters included as to to whom they were sent and on which date. Further to this an admin will have the ability to keep track of which new admins were added to the system, by who and on which date.

- Technical specifications

    - All reports will mask out sensitive information during display even when viewed by an admin.
    - If the report generation is too time consuming which is a possibility if the selected time gap is quite large, the generation will be added to a queue and the user will be automatically notified through an alert once it has been successfully generated.

Literature Review

As with any good system, prior to implementing Watchnode there was a background search on the competitor market for similar systems and the criteria based on which it was done were as follows:

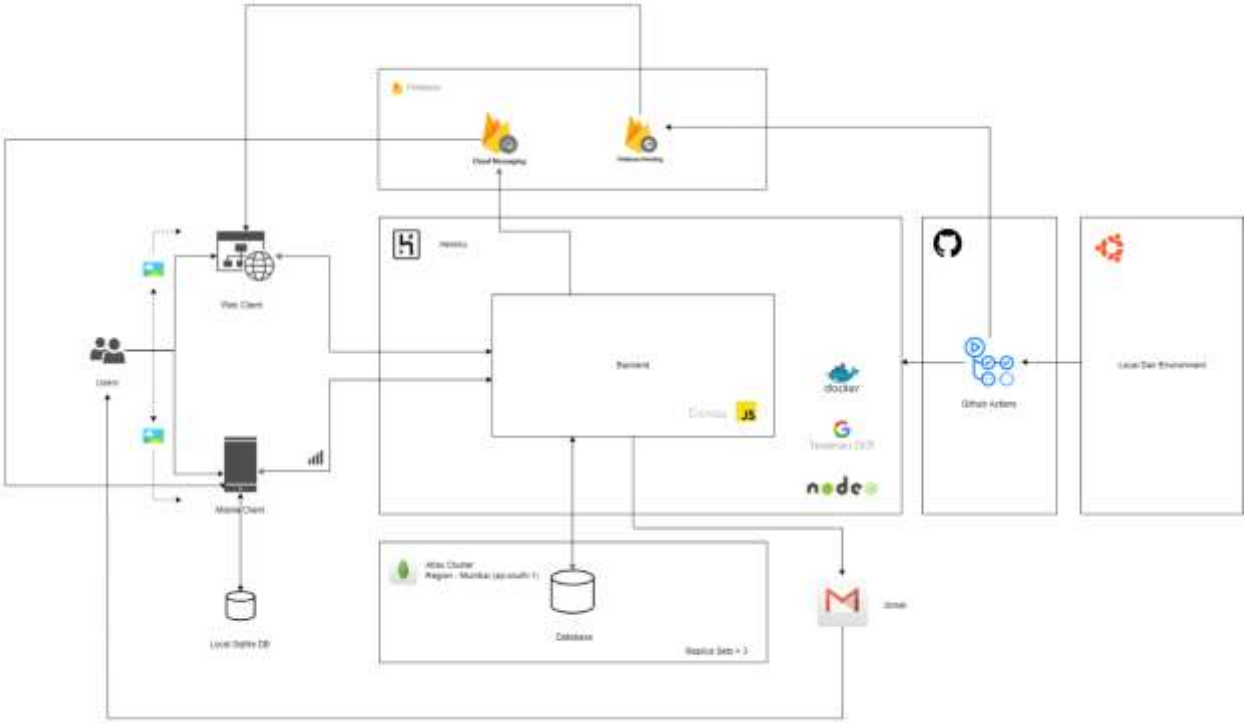
- Schedule recognition
- Schedule management
- Automated alerts
- Offline functionality

Upon a thorough analysis, while there are quite a few systems which provide the last three, a system capable of uploading and digitalizing custom schedules was nonexistent. Accompanying that while many do provide offline functionality the time period for which it is available is limited.

System developed by huge tech giants such as Google and Microsoft and their products Google Calendar and Microsoft Outlook calendar fall into the same space as Watchnode. While it's true that Watchnode does indeed have an edge over them due to its feature of custom schedule uploads, it's still missing out on major functionalities provided by the former mentioned systems which is why we have added integrations with both to our roadmap. Both provide a set of REST endpoints which could be used to sync data between the platforms. Further to this

Watchnode will include multi language support and multiple theme support in the future as planned in the roadmap.

System Architecture



Watchnode will follow the above system architecture during the development stage where both the web and the mobile client will communicate with a separate server hosted in Heroku. The free plan of Heroku provides considerable dyno hours to ensure a smooth run through the development period. It also provides a container registry to which the backend application can be dockerized and pushed during the deployment stage. Heroku The frontend on the other hand will be hosted on Firebase. The system will use an Atlas Cluster as its primary database, Gmail as the email service provider and Firebase cloud messaging for push notifications. GitHub actions will facilitate the CI/CD pipelines of the system. There will be 2 types of jobs in these pipelines. The test jobs would be triggered on opening pull requests to the development branch and the deployment jobs will be triggered on pushes or merges to development or master. This together with our Git branching strategy will ensure a smooth flow in the system.

To consider additional security and smoother management the following steps will be ensured in the backend

- All important events happening in the backend at the API level will be logged for future references. These logs will be strictly masked and will not expose any sensitive information
- An audit log configured in the database will keep track of all modifications which happen to the database
- All keys and other sensitive information will be excluded from source control and maintained as environment variables in the deployment environments
- Rate limiters and payload sanitizers configured at the backend level will reduce the threat of DOS and XSS attacks.
- CORS will be enabled in the backend only for the domains where the frontend will be hosted

References

- https://github.com/tesseract-ocr/tesseract
- https://firebase.google.com/docs/cloud-messaging
- https://firebase.google.com/docs/hosting
- https://devcenter.heroku.com/articles/container-registry-and-runtime
- https://github.com/marketplace/actions/deploy-docker-image-to-heroku-app
- https://developers.google.com/calendar/api
- https://docs.microsoft.com/en-us/graph/outlook-calendar-concept-overview
- https://www.mongodb.com/docs/manual/core/auditing/