41113 Software Development Studio

# Project Brief

# Pyrmont Action Website Upgrade

PYRMONTACTION

UTS


| Student Name | Student ID |
|---|---|
| Anthony He | 13869728 |
| Charbel Moubarak | 25126689 |
| Henil Patel | 24505289 |
| Aaryan Mahajan | 25407192 |
| Massimo Fathulla | 24945545 |

## Table of Contents

# 1 Project Overview

## 1.1 The Client

Pyrmont Action is a not-for-profit incorporated community association that works to improve development in Pyrmont for the benefit of the community. It collaborates with the community, councils, and government to ensure the voices of locals are heard.

## 1.2 Problem Statement

Currently, Pyrmont Action does not have a functioning website that meets its needs. The organisation lacks a dedicated account management feature, making it difficult for both community members and administrators to navigate. In addition, there is no secure way for visitors to join as members and complete payment transactions. Another issue is that private, administrator-facing content is not adequately protected and has been exposed.

## 1.3 Project Goals and Objectives

The goal is to develop a polished and fully functional website for Pyrmont Action. This will be achieved by combining the two previously developed code bases and implementing additional features. The new platform will draw on the strengths of both code bases, providing a solid foundation for further development. The primary focus will be on account management, ensuring appropriate access controls for authorised accounts. Membership payments will be handled through the Stripe platform, as preferred by the client. Finally, additional public-facing features not included in the earlier code bases will be implemented to make the website more engaging and useful for the community.

## 1.4 Project Timeline

### 1.4.1 Sprint Planning

| ***Sprint Number*** | **Sprint Period** | **Milestones and Deliverables** | **Name of Features/Functions** | **Associated User Stories** |
|---|---|---|---|---|
| *0* | *25/09 – 08/09* | *Milestone:*<br>● *Project Initiation*<br>*Deliverable:*<br>● *Meeting Time Schedule* | | |
| *1* | *08/09 – 15/09* | *Milestone:*<br>● *Agreeing with the actionable requirements*<br>● *Previous codebases running successfully*<br>*Deliverable:*<br>● *Project Brief* | | |
| *2* | *15/09 – 29/09* | *Milestone:*<br>● *Successful in merging the two code bases.*<br>*Deliverable:*<br>● *Database Setup*<br>● *User Authentication*<br>● *Stripe Integration* | ● Onboarding<br>● Account Management<br>● Meeting Minutes<br>● Payment Integration | ● U101 to U105<br>● U201 to U211<br>● U301 to U306<br>● U401 to U406 |

| | | | | |
|---|---|---|---|---|
| | | • *Document Uploading*<br>• *Account CRUD* | | |
| 3 | *29/09 – 13/10* | *Milestone:*<br>• *Successful implementation of assigned features*<br>• *Documentation*<br>*Deliverable:*<br>• *Editorial CRUD*<br>• *Solution Architecture Documentation*<br>• *Public-facing content*<br>• *Private-facing content* | • Editorial Dashboard<br>• Event Calendar<br>• Member List Viewer | • U501 to U505<br>• U601 to U606<br>• U701 to U706 |
| 4 | *13/10 – 27/10* | *Milestone:*<br>• *Complete all features*<br>• *Finalise Testing*<br>• *Documentation*<br>*Deliverable:*<br>• *Handover documentation*<br>• *Acceptance Criteria Checklist* | • Blog Submission and Approval<br>• Contact Us Form Email Integration | • U801 to U808<br>• U901 to U903 |

### 1.4.2 Communication and Collaboration Plan

#### *1.4.2.1 Communication Goals*

The primary goal of communication within the team is to ensure all members are aligned on the project's current status, upcoming tasks, and any challenges that may arise. Effective communication will help maintain productivity, accountability, and collaboration throughout the project lifecycle.

Communication with the client is equally important. Weekly updates will be provided to keep the client informed of progress, gather feedback, and ensure the project remains aligned with their expectations.

**Communication Platforms:**

- Team Communication:
  - Primary Platforms - Discord (for daily updates and quick discussions) and in-person meetings (for deeper collaboration and planning).
  - Other Platforms - Zoom or Microsoft Teams may be used if meetings are to be recorded.

- Client Communication:
  - Weekly meetings will be conducted via Zoom or Microsoft Teams, depending on client preference.
  - Meeting agendas and progress reports will be shared in advance to facilitate productive discussions and timely feedback.

#### *1.4.2.2 Communication Plan*

| ***Student*** | **Role** | **Contact** | **Communication Responsibility Frequency** | **Format/Channel** | **Notes** |
|---|---|---|---|---|---|
| *Massimo Fathulla* | Team Leader/SCRUM Master | Massimo.E.Fathulla@student.uts.edu.au | Daily, weekly | Facilitates daily in-person Scrum meetings, provides weekly progress reports using a project management tool, and sends monthly high-level updates on timeline, budget and progress to the product owner via email. Additionally, it is responsible for | Primary point of contact for issue resolution and escalation. |

| | | | | | |
|---|---|---|---|---|---|
| | | | | documentation and overall project management. | |
| *Aaryan Mahajan* | Frontend Developer | Aaryan.Mahajan@student.uts.edu.au | Daily, Weekly | Attends stand-up meetings, client meetings and Scrum meetings, communicates with the client and backend developers, and designs wireframes for feedback. | Responsible for translating user needs into frontend functionality. |
| *Anthony He* | Software Architect / Backend Developer | Anthony.Z.He@student.uts.edu.au | Daily, Weekly | Attends stand-up meetings, client meetings and Scrum meetings, assists in defining project requirements, and communicates with the client, team leader and frontend developer to determine software needs. | Provides technical guidance and backend solutions. |
| *Henil Patel* | Unit Tester / Quality Assurance | Henil.Patel@student.uts.edu.au | Daily, Weekly | Attends stand-up meetings, client meetings and Scrum meetings, communicates with the client, frontend and backend developers, and provides regular quality assurance updates. | Ensures deliverables meet acceptance criteria before deployment. |
| *Charbel Moubarak* | Backend Developer | Charbel.Moubarak@student.uts.edu.au | Daily, Weekly | Attends stand-up meetings, client meetings and Scrum meetings, communicates with the client, team leader and other backend developers. | Supports backend tasks, bug fixes, and API integration. |

### *1.4.2.3 Collaboration Plan*

| ***Type*** | **SCRUM of SCRUM** |
|---|---|
| *When/Where/Participants* | Held in person around the Scrum board for a 15-minute stand-up meeting with all team members. |
| *Share* | • Each person reports on tasks in progress, tasks completed, and tasks upcoming. |

- Problems are raised.
- Requests for assistance are made if required.

| *Type* | **Weekly Scrum/Standup Meeting** |
| --- | --- |
| *When/Where/Participants* | Conducted either online or in person for 15–30 minutes, attended by all team members. |
| *Share* | • Each person reports on tasks in progress, tasks completed, and tasks upcoming.<br>• Issues are flagged.<br>• Requests for help are raised where necessary. |

| *Type* | **Weekly/Fortnightly Client Meeting** |
| --- | --- |
| *When/Where/Participants* | Held online or in person with the client and relevant team members, lasting 15–30 minutes. |
| *Share* | • Present project progress and deliverables.<br>• Confirm or clarify any new requirements. |

| *Type* | **Discord/Teams Channel** |
| --- | --- |
| *When/Where/Participants* | Open real-time communication and file sharing space, accessible to all team members. |
| *Share* | • Burndown chart updates.<br>• Project documentation.<br>• General questions and discussions for team visibility. |

| *Type* | **Sprint Planning and Retrospective** |
| --- | --- |
| *When/Where/Participants* | At the start and end of each two-week sprint, held online or in person with all Developers, Quality Assurance, and the Product Owner. |
| *Share* | • **Share two days in advance:**<br>◦ Agenda for meeting<br>◦ Product backlog<br>◦ QA report<br>◦ Attendees needed for meeting (includes product owner)<br>◦ Budget |

- **Meeting format:**
    - Agenda review
    - Review product backlog
    - Questions/Discussions
    - Next steps review
- **Email/Message (immediately after meeting):**
    - Meeting notes to all attendees
    - Sprint goals

| ***Type*** | **Weekly/Monthly Emails** |
|---|---|
| *When/Where/Participants* | Prepared by the team leader and sent to the product owner. |
| *Share* | • Progress against the project plan.<br>• Support required.<br>• Issues or risks identified.<br>• Updated timeline.<br>• Upcoming activities. |

## 1.5 Quality/Success Criteria

**Functional Requirements**

- New users can successfully register, pay the $25 annual membership fee, and gain member access without errors.
- Role-based access is enforced: Community Members, Content Managers and Administrators only see the features relevant to their role.
- Administrators can upload and publish meeting minutes; members can securely view/download them.
- Administrators/Content Managers can add or update content (e.g., events, blogs) through the editorial dashboard.
- The “Contact Us” form reliably delivers enquiries to the designated email.

**Non-Functional Requirements**

- The application loads key pages within three seconds under normal use.
- The web app is accessible on modern browsers (Chrome, Firefox, Safari, Edge) and mobile browsers.
- The system maintains at least 95% uptime during demonstration and evaluation.

**Testing & Reliability**

- All core features pass unit and integration testing (e.g., onboarding flow, payments, CRUD operations).
- Stripe payment testing demonstrates consistent success in transactions.
- Quality assurance reports show no critical defects outstanding at handover.

**Usability & Client Satisfaction**

- Users across all access levels can navigate the website without major issues (validated in user acceptance testing).
- The client confirms the website meets the agreed requirements and is suitable for ongoing use by Pyrmont Action.
- Documentation is sufficient for administrators to manage content and membership independently after handover.

## 1.6 Project Scope

### 1.6.1 In scope

| Ref. | Item | Description |
|---|---|---|
| SC001 | Web Application | Develop a responsive web application for the Pyrmont Action group. |
| SC002 | Membership Payment | • Allow users to pay a $25 annual membership fee when joining.<br>• Provide functionality for members to renew their membership before expiry.<br>• Integrate the Stripe platform for secure payment processing. |
| SC003 | Three Access Levels | • Community Member<br>  ○ Can download meeting minutes.<br>  ○ Can view their membership duration and payment status.<br>• Administrator<br>  ○ Manage user accounts.<br>  ○ Publish meeting minutes.<br>  ○ Edit content with appropriate permissions.<br>• Content Managers<br>  ○ Have permission to edit content only.<br>  ○ Do not have access to administrative functions. |
| SC004 | Editorial Dashboard | • Administrators and Content Managers can edit existing public-facing content.<br>• Public-facing web pages are listed in the dashboard.<br>• Editors can select a page and edit items within it.<br>• Create, Read, Update and Delete (CRUD) operations are available for all page items. |
| SC005 | New public-facing content | • Public blog submissions are enabled, with Administrators approving or rejecting posts.<br>• An event calendar is visible to the public and editable by Administrators.<br>• A simple email integration supports the Contact Us form, sending enquiries to the authorised email address. |

### 1.6.2 Out of Scope

| Ref. | Item | Description |
| --- | --- | --- |
| OC001 | Mobile Application | Building a standalone mobile app is excluded; the web app will be responsive instead. |
| OC002 | Post-project Work | Any maintenance or enhancements beyond the project duration are excluded. |
| OC003 | Marketing Email Tools | Bulk newsletters, advanced marketing automation or campaign tools are excluded; only Stripe receipts and basic admin/member notifications are included. |

## 1.7 Assumptions

1. Each person within the team has the ability and willingness to complete the tasks assigned to them prior to the set deadlines
2. Team members and clients will be available for our weekly meetings
3. Scope is thoroughly discussed and agreed upon by both the team members and the client
4. The architecture chosen to use for all aspects of the project will be rational and stable for the entire project duration

## 1.8 Risks

**Schedule Issues:** As the project timeline is limited, issues may arise in regards to the scheduling and completion of work, leading to the team potentially falling behind our planned schedule.

**Scope Creep:** Extra features may be requested during the project lifecycle, impacting the projected timeline.

**Lack of Testing:** Throughout the project's lifecycle, if testing is not properly implemented, it can potentially lead to some serious issues later in the project.

## 1.9 Issues

**Late Start:** As we started this project later than would be normal for this subject, this has caused numerous issues to our development timeline. We missed certain crucial milestones in the early stages of the project and so have needed to catch up on them.

## 1.10 Constraints

**Time:** As the project timeline is based on a university semester, this limits the project duration.

**Busy Schedules:** All team members will be balancing this subject with their other subjects and commitments they have.

**Scope:** The scope will define the features that will be designed and implemented for the site.

**Resources:** The resources and framework that the client has access to, and

## 1.11 Dependencies

**Hosting and Domain Setup**: Client must confirm arrangements for website hosting and domain.

**MongoDB Atlas + Mongoose**: Required for cloud database and schema management.

**Stripe API (Checkout + Webhooks)**: Dependency for secure payment processing and membership management.

**Nodemailer (dummy Gmail for development)**: Dependency for sending admin/member email notifications.

**Testing Frameworks**: Playwright (End-to-End) and Mocha/Chai (Unit Testing) required for automated testing.

**Documentation & Version Control**: GitHub repository for source code collaboration; team must provide adequate project documentation for transparency.

**Client Communication**: Ongoing collaboration with the client to validate requirements and scope changes.

# 2 Requirements

## 2.1 Functional Requirements

| Function /Feature/ Task | Priority | Difficulty | Complexity | Testing | Definition of Done | Resources |
|---|---|---|---|---|---|---|
| | *(Scale between 1-5, where 1 is the lowest and 5 is the highest)* | | | *Identify and describe how and when the product will be tested over the course of the project/semester.* | *Describe a set of conditions that allow your team to assess the product's completion for the project/semester.* | *List the required library, API, language, package, connected function(s), connected database etc.* |
| Community Member Onboarding | 5 | 3 | 3 | Unit test account creation, integration test full onboarding flow | A visitor can become a Community Member. | MongoDB Atlas, Mongoose, Node.js, Express, Mocha & Chai for unit testing |
| Account Management | 5 | 3 | 3 | Unit test CRUD actions, integration test role updates | User can log in/out, update details, deactivate account, and role changes propagate correctly | MongoDB Atlas, Mongoose, Mocha & Chai for unit testing, JWT and bcrypt |
| Membership Payment Integration | 5 | 4 | 4 | Simulate Stripe test payments, verify webhook receipts | Payment successfully processed, receipt emailed, membership marked as paid | Stripe API (Checkout + Webhooks), Node.js , Postman for API testing |
| Meeting Minute Access | 4 | 2 | 2 | Upload/download test, permission checks | An Administrator can upload and publish Meeting Minutes, and | MongoDB, HTML, Node.js |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | Community Members can view them. | |
| Editorial<br><br>Dashboard | 4 | 4 | 4 | CRUD content test, image upload test | A Content Manager or Administrator can make changes to the public content. | Vue, Node.js, MongoDB |
| Member List Viewer | 4 | 2 | 2 | Query test, filter test | Administrators can view details of all members on the list. | MongoDB, Mongoose |
| Event Calendar | 3 | 3 | 3 | CRUD event test, calendar UI test | Administrators can create events visible to Visitors. | Vue, MongoDB, Mocha & Chai for Unit testing, Apache JMeter |
| Blog Submission and Approval | 3 | 3 | 3 | Submission, approval/rejection, publish visibility tests | A Visitor can submit and an Administrator can approve. | MongoDB, Vue, Node.js, Express, Postman for API testing |
| Contact Us Form Email Integration | 2 | 2 | 2 | Form validation test, email send test | The authorised email receives the Visitor’s enquiry. | Nodemailer (Node.js module for sending emails), Mocha & Chai for Unit testing |

## 2.2 Non-Functional Requirements

| Function /Feature/ Task | Priority | Difficulty | Complexity | Testing | Definition of Done | Resources |
|---|---|---|---|---|---|---|
| | *(Scale between 1-5, where 1 is the lowest and 5 is the highest)* | | | *Identify and describe how and when the product will be tested over the course of the project/semester.* | *Describe a set of conditions that allow your team to assess the product's completion for the project/semester.* | *List the required library, API, language, package, connected function(s), connected database etc.* |
| Performance | 5 | 3 | 3 | Load testing (50 concurrent users), response time monitoring | System responds under 2 seconds with 50 concurrent users | Apache JMeter, Chrome DevTools |
| Security | 5 | 4 | 4 | Penetration testing, JWT expiry checks | All pages behind login require a valid JWT, passwords are hashed, and sensitive data is encrypted | bcrypt, JWT, Helmet.js, TLS |
| Usability | 4 | 2 | 2 | User acceptance tests, UI/UX review | ≥80% of users complete key tasks without guidance and report ease of use | Figma (prototyping), user test participants, |
| Accessibility | 3 | 2 | 2 | Screen reader test, WCAG checklist | All pages meet WCAG 2.1 AA guidelines | axe DevTools, Lighthouse |
| Backup and Recovery | 2 | 2 | 2 | Database backup/restore simulation | Weekly backups run successfully, recovery completed within 2 hours | MongoDB Atlas backups |

## 2.3 User Stories/Narratives

Note:

**Priority:** H = High, M = Medium, L = Low

**Estimate:** 1 is the lowest effort, 10 is the highest effort

### 2.3.1. Community Member Onboarding

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U101 | As a Visitor, I want to locate and press the “Join” button so that I can start the onboarding process to become a Community Member. | H | 4 |
| U102 | As a Visitor, I want to provide my personal information when becoming a member, so that I can progress in the onboarding process. | H | 6 |
| U103 | As a Visitor, I want to securely create a password so that I can access my account in the future. | H | 5 |
| U104 | As a Visitor, I want to provide my payment details so that I can complete the membership contribution during the onboarding process. | H | 6 |
| U105 | As a Visitor, I want to be redirected to the Login page so that I can immediately access my account after onboarding. | L | 3 |

### 2.3.2. Account Management

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U201 | As an Account Holder, I want to locate and press the “Login” button, so that I can access my account. | H | 4 |
| U202 | As an Account Holder, I want to log in using my email and password so that I can access the correct account. | H | 6 |
| U203 | As an Account Holder, I want my password to be masked so that it is not displayed when in public. | M | 1 |
| U204 | As an Account Holder, I want to update my personal details so that my account information stays accurate. | M | 4 |
| U205 | As an Account Holder, I want to deactivate my account so that all my data is removed from the system. | M | 2 |
| U206 | As an Account Holder, I want to change my password so that I can strengthen the security of my account. | M | 2 |
| U207 | As an Account Holder, I want to update my email address so that notifications are forwarded to my preferred communication method. | M | 2 |
| U208 | As an Administrator, I want to create a Content Manager account so that they can edit the public-facing content. | H | 5 |
| U209 | As an Administrator, I want to create an Administrator account so that they can access everything on the system. | H | 5 |

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U210 | As an Administrator, I want to deactivate an Account Holder so that they lose system access and permissions. | M | 2 |
| U211 | As an Administrator, I want to change the role of an Account Holder so that permissions are enforced correctly. | L | 4 |

### 2.3.3. Membership Payment Integration

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U301 | As a Visitor or Community Member, I want to be redirected to the payment platform (Stripe) so that I can process my payment. | H | 7 |
| U302 | As a Visitor, I want to use the Stripe payment platform, so that I can securely process my payment with Pyrmont Action. | H | 8 |
| U303 | As the Treasurer, I want to be sent email notifications when a payment is processed so that I can update financial records. | H | 6 |
| U304 | As a new or current Community Member, I want to receive an automatic receipt by email after payment so that I have proof of membership. | H | 6 |
| U305 | As a Community Member, I want to be notified when my membership is about to expire so that I can renew on time. | M | 4 |
| U306 | As a Community Member, I want my membership status and renewal date shown so that I am aware when to renew it. | L | 3 |

### 2.3.4. Meeting Minute Access

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U401 | As an Administrator, I want to upload Meeting Minutes as a PDF on a private-facing webpage so that members can access them securely. | H | 6 |
| U402 | As an Administrator, I want to control when Meeting Minutes are published so that only finalised documents are visible to members. | H | 4 |
| U403 | As a Community Member, I want to view published Meeting Minutes so that I can stay informed about decisions and updates. | H | 3 |
| U404 | As an Administrator, I want to edit or replace uploaded Meeting Minutes so that I can correct errors or update content. | M | 5 |
| U405 | As a Community Member, I want to receive a notification when new Meeting Minutes are published so that I can stay updated. | M | 4 |
| U406 | music | L | 5 |

### 2.3.5. Editorial Dashboard

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U501 | As an Administrator or Content Manager, I want to access the editorial dashboard so that I can manage public-facing webpages in one place. | H | 7 |
| U502 | As an Administrator or Content Manager, I want to see a list of all editable webpages so that I can quickly navigate to the one I need. | H | 5 |
| U503 | As an Administrator or Content Manager, I want to select a webpage from the dashboard so that I can access editing features for it. | H | 3 |
| U504 | As an Administrator or Content Manager, I want to upload images and text to a webpage so that I can enrich the content. | M | 6 |
| U505 | As an Administrator or Content Manager, I want to remove outdated items from a webpage so that the content stays relevant. | M | 4 |

### 2.3.6. Member List Viewer

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U601 | As an Administrator, I want to view a list of all registered Community Members so that I can monitor membership. | M | 7 |
| U602 | As an Administrator, I want each member's full name to be displayed so that I can identify individuals. | M | 2 |
| U603 | As an Administrator, I want each member's email address to be shown so that I can contact them if needed. | M | 2 |
| U604 | As an Administrator, I want to see whether a member is active or inactive so that I can track renewals. | M | 3 |
| U605 | As an Administrator, I want to filter the list by active/inactive status so that I can manage renewals efficiently. | L | 6 |
| U606 | As an Administrator, I want to search for a member by name or email so that I can locate specific records. | L | 6 |

### 2.3.7. Event Calendar

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U701 | As an Administrator, I want to add new events to the Event Calendar so that the community can stay informed. | H | 6 |
| U702 | As a Visitor, I want to view upcoming events on the Event Calendar so that I can participate in community activities. | H | 4 |
| U703 | As an Administrator, I want to include the event name, date, location, and description so that all relevant details are visible. | H | 6 |
| U704 | As an Administrator, I want to edit event details after posting so that I can correct or update information. | M | 7 |
| U705 | As an Administrator, I want to delete past or cancelled events so that the calendar stays clean and accurate. | M | 5 |
| U706 | As a Visitor, I want to click on an event to see full details so that I can decide whether to attend. | L | 4 |

### 2.3.8. Blog Submission and Approval

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U801 | As a Visitor, I want to write and submit blog content so that I can express my views to the community. | M | 5 |
| U802 | As a Visitor, I want to draft a blog post anonymously so that I can share my thoughts without revealing my identity. | M | 5 |
| U803 | As a Visitor, I want to draft a blog post with my name so that I can be credited for my contribution. | M | 5 |
| U804 | As a Visitor, I want to be prompted to review my draft before submitting it, so that I can fix grammatical errors and improve clarity. | M | 5 |
| U805 | As an Administrator, I want to view all submitted blog drafts so that I can manage approvals. | H | 7 |
| U806 | As an Administrator, I want to reject inappropriate blog posts so that the site remains respectful and safe. | H | 7 |
| U807 | As an Administrator, I want to edit blog posts before publishing so that I can fix formatting or minor issues. | L | 7 |
| U808 | As a Visitor, I want to view published blog posts so that I can read community contributions. | H | 6 |

### 2.3.9. Contact Us Form Email Integration

| ID | User Story | Priority | Estimate |
|---|---|---|---|
| U901 | As a Visitor, I want to submit a message through the contact form so that I can reach Pyrmont Action. | M | 3 |
| U902 | As an Enquiry Email Holder, I want to receive the Visitor's message, name, email address, and subject line via email so that I can respond appropriately. | L | 3 |
| U903 | As an Enquiry Email Holder, I want the contact form to be configured to send emails to a specific email address so that enquiries reach the correct recipient. | L | 3 |

## 2.4 User Narratives

### 2.4.1. Account Management

| | |
|---|---|
| Use Case ID | UC206 |
| User Story | As an Account Holder, I want to change my password so that I can strengthen the security of my account. |
| Goal | The Account Holder updates their password to a more secure one. |
| Priority | Medium |
| Actors | Primary Actor: Account Holder<br>Secondary Actors: Database |
| Pre-conditions | The Account Holder is logged in. |
| Post-conditions | Account details are updated in the database. |
| Trigger | The Account Holder selects "My Account" from the navigation bar.<br>The Account Holder initiates a password change. |
| Main Flow | 1. The Account Holder presses the login button.<br>2. They enter their account credentials and log in. If the email or password is wrong, refer to **Alternate Flow 1**.<br>3. They navigate to their account page and view their account management option.<br>4. They select "change password" and enter their current password with their new password. If any of the old passwords are wrong, refer to **Alternate Flow 2**.<br>5. They click "confirm".<br>6. They click Save.<br>7. The system validates and stores the updates.<br>8. The system confirms success.<br>9. The use case ends. |
| Exceptions | E1. Invalid field inputs (e.g. weak password, duplicate email).<br>E2. Session timeout during editing. |
| Includes | UC201: Login from the front page<br>UC202: Log in with account credentials |
| Supporting Information | All personal data must comply with the organisation's privacy policy. |
| Non-functional Requirements | Performance: Average update < 2 seconds.<br>Security: Passwords are masked when logging in. |

| Alternate Flow 1 | "Wrong Password" |
|---|---|
| **Trigger** | The Account Holder used the wrong password to log in. |
| **Step** | 1. The login webpage will display an error message.<br>2. The Account Holder enters their password again.<br>3. Re-join at Step 2 in the Main Flow. |
| **Alt Flow** | N/A |
| **Post conditions** | The Account Holder can successfully log in. |
| **Exceptions** | |

| Alternate Flow 2 | "Incorrect current password" |
|---|---|
| **Trigger** | The Account Holder entered the wrong current password |
| **Step** | 1. The webpage will display an error message.<br>2. The Account Holder enters their password again.<br>3. Re-join at Step 4 in the Main Flow. |
| **Alt Flow** | N/A |
| **Post conditions** | The Account Holder can successfully change their password. |
| **Exceptions** | |

### 2.4.2. Membership Payment Integration

| Use Case ID | UC303 |
|---|---|
| User Story | As the Treasurer, I want to be sent email notifications when a payment is processed so that I can update financial records. |
| Goal | The Treasurer is emailed the receipt of a new membership payment. |
| Priority | High |
| Actors | Primary Actor: Treasure<br>Secondary Actors: Stripe API, Visitor |
| Pre-conditions | - The Visitor is onboarding to become a Community Member.<br>- The Stripe payment system is active. |
| Post-conditions | - Payment is processed and recorded.<br>- A receipt is emailed to the Treasurer.<br>- A receipt is emailed to the Community Member. |
| Trigger | The Community Member processes payment transactions for membership. |
| Main Flow | 1. The Community Member clicks the add payment option.<br>2. The system redirects to the Stripe checkout page.<br>3. The Member enters payment details and confirms. Refer to **Alternate Flow 1.**<br>4. Stripe processes the payment and returns success.<br>5. The system updates the membership status.<br>6. The system emails a receipt.<br>7. The use case ends. |
| Exceptions | E1. Card declined or payment error.<br>E2. Webhook confirmation delayed. |
| Includes | U302 - Membership Payment<br>U104 - Onboarding Payment |
| Supporting Information | The payment must meet PCI DSS security standards. |
| Non-functional Requirements | Performance: End-to-end < 3 seconds after Stripe confirmation. |

| Alternate Flow 1 | "Card Declined" |
|---|---|
| **Trigger** | The Stripe platform denies the card of the owner. |

| | |
|---|---|
| **Step** | 1. The system shows a “pending” banner and a retry status in the background<br>2. When the webhook arrives, the system updates the membership and emails the receipt.<br>3. The payment is recorded correctly<br>4. Re-join at Step 3 in the Main Flow. |
| **Alt Flow** | N/A |
| **Post conditions** | The payment is recorded correctly. |
| **Exceptions** | E1. Step 2 – Webhook never arrives → flagged for manual review. |

### 2.4.3. Editorial Dashboard

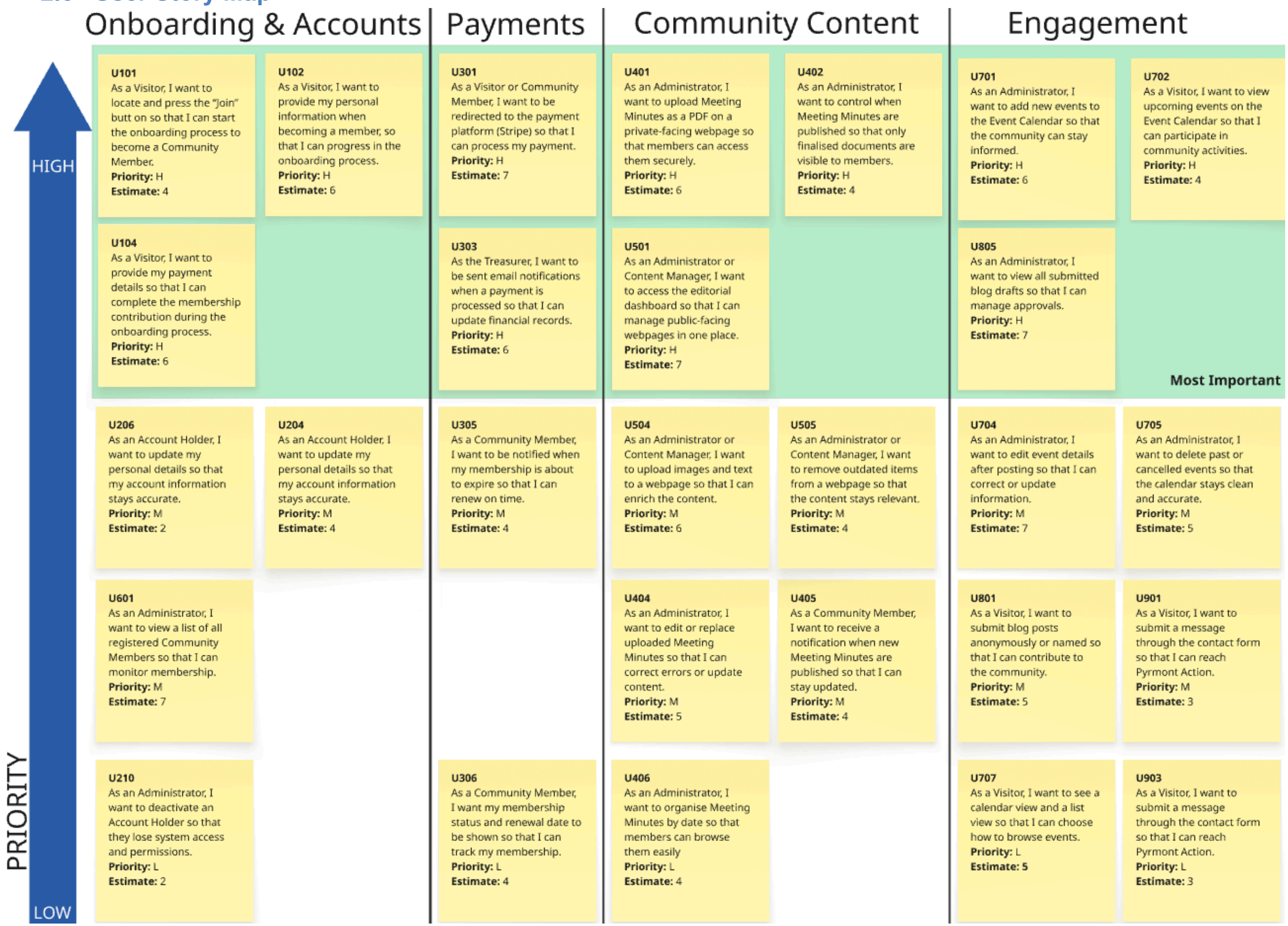
| | |
|---|---|
| Use Case ID | UC501 |
| User Story | As an Administrator or Content Manager, I want to edit pages so that content stays current. |
| Goal | The authenticated User can edit and publish content to the public-facing side. |
| Priority | High |
| Actors | Primary Actor: Administrator or Content Manager<br>Secondary Actors: Database, Web server |
| Pre-conditions | The Administrator/Content Manager is logged in. |
| Post-conditions | The changes are saved and appear live on the website. |
| Trigger | The user selects a webpage from the editorial dashboard. |
| Main Flow | 1. The user opens the editorial dashboard.<br>2. The user selects a webpage tile. **Refer to Alternate Flow 1**<br>3. The system shows the editing interface.<br>4. The user updates text, images, or layout.<br>5. The user clicks Publish. **Refer to Alternate Flow 2**<br>6. The system updates the live content.<br>7. The use case ends. |
| Exceptions | E1. Invalid image file format.<br>E2. Internet connection lost during save. |
| Includes | UC504: Upload Content |
| Supporting Information | Version history should be stored for all edits. |
| Non-functional Requirements | Performance: Update reflects within 2 seconds.<br>Maintainability: All changes are logged with the editor's name and time. |

| | |
|---|---|
| **Alternate Flow 1** | “Edit Conflict” |
| **Trigger** | Two instances of editing operations on the same page simultaneously. |
| **Step** | 1. The system detects a conflict and blocks the second save.<br>2. The system notifies the second attempt to enter the editing page.<br>3. The second attempt is denied.<br>4. Re-join at Step 3 in the Main Flow. |
| **Alt Flow** | |
| **Post conditions** | The first instance of editing is not interfered with. |
| **Exceptions** | E1. Step 3 – Conflict detection fails and overwrites changes |

| | |
|---|---|
| **Alternate Flow 1** | “Save as Draft” |
| **Trigger** | The Editor chooses to save without publishing. |
| **Step** | 1. The Editor leaves the webpage. |

| | |
|---|---|
| | 2. The content is stored as a draft on the dashboard, public to all Editors.<br>3. The draft is shown as an item with a special tag in the editable items list.<br>4. The user can later publish it.<br>5. Re-join at Step 5 in the Main Flow. |
| **Alt Flow** | |
| **Post conditions** | The content is not visible publicly. |
| **Exceptions** | E1. Step 1 – Draft is lost due to a browser crash. |

## 2.5 User Story Map



## 2.6 Other notes

# 3 Sign Off

## 3.1 Signature of Team Members:

Name: Henil Patel Signature: H.P.

Name: Anthony He Signature: A.H.

Name: Aaryan Mahajan Signature: A.M.

Name: Massimo Futhalla Signature: M.F.

Name: Charbel Moubarak Signature: C.M.

## 3.2 Signature of Client:

Name: ______________ Signature: ______________________

Name: ______________ Signature: ______________________

Name: ______________ Signature: ______________________